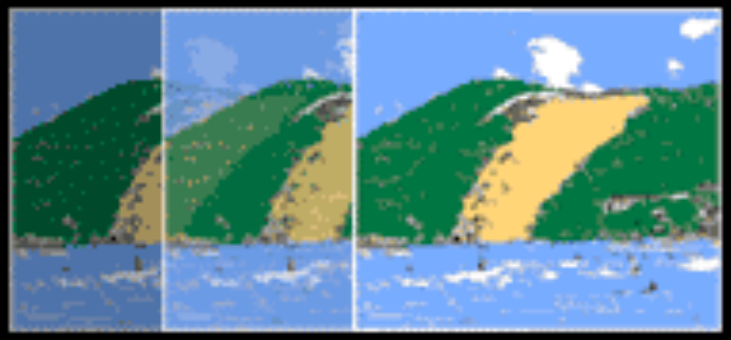
GPU-Based Volume Rendering of Unstructured Grids

Module 4:
Hardware-Assisted Visibility Sorting

Steven P. Callahan

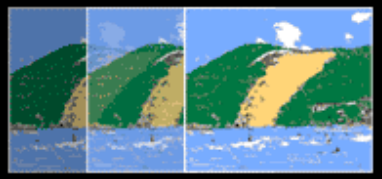
University of Utah



SIBGRAPI 2005

Natal - RN - Brazil

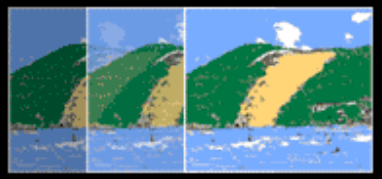
XVIII Brazilian Symposium on Computer Graphics and Image Processing



SIBGRAPI 2005

Overview

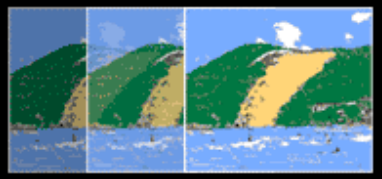
- Recent advances in GPU programmability
- *k*-Buffer: A fragment stream sorter
- Hardware-Assisted Visibility Sorting
- Dynamic Level-of-Detail



GPU: Recent Features

Render to texture

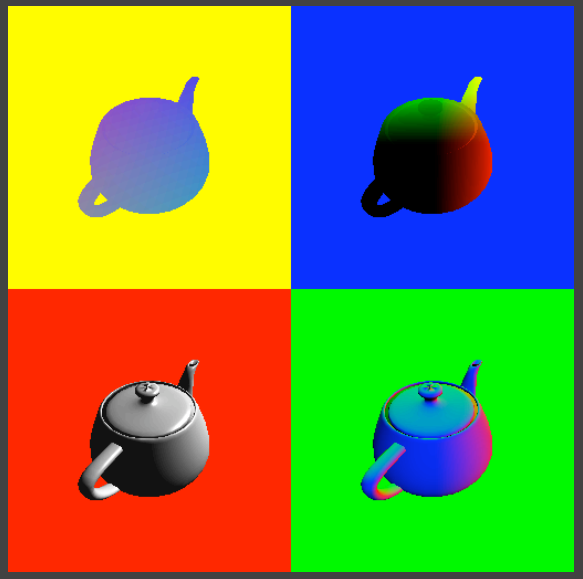
- Why?
  - Better performance
- Applications
  - Dynamic textures
  - Multi-pass algorithms
  - Image processing



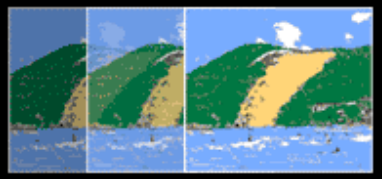
GPU: Recent Features

Multiple Render Targets (MRTs)

- Write into multiple textures simultaneously



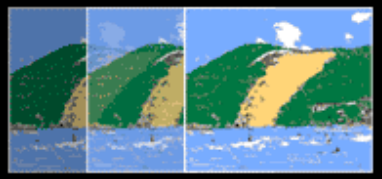
[NVIDIA]



GPU: Recent Features

OpenGL Pixel Buffers (PBuffers)

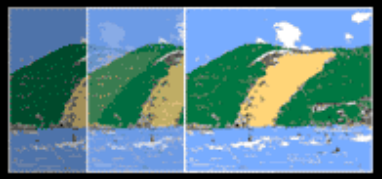
- Enables off-screen rendering
- Contains its own depth, stencil, and aux buffers
- MRT support by rendering into Front and up to 3 AUX buffers



GPU: Recent Features

Disadvantages of PBuffers

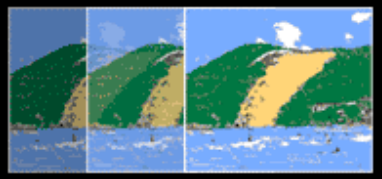
- Each has its own OpenGL context
- Switching between PBuffers is expensive
- Cannot share buffers between PBuffers
- Pixel format selection
- Extensions only available on Windows



GPU: Recent Features

OpenGL Framebuffer Objects (FBOs)

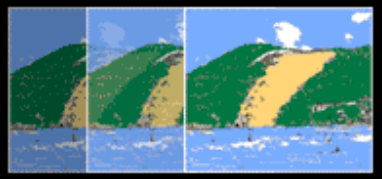
- A collection of attachable textures
  - Color, depth, stencil, etc.
- Attached textures are source and destination for fragment shaders
- MRTs are available using multiple color attachments



GPU: Recent Features

Advantages of FBOs

- A single context
- Pixel format determined by texture format
- Share buffers between FBOs
- Easier to use than PBuffers
- Works on multiple platforms

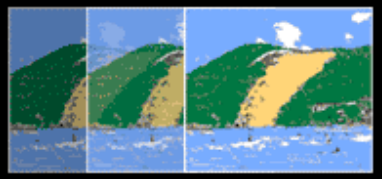


GPU: Recent Features

Code

- PBuffers
  - RenderTexture 2.0 (Mark Harris)
- FBOs
  - Framebuffer Object Class (Aaron Lefohn)

www.gpgpu.org/developer

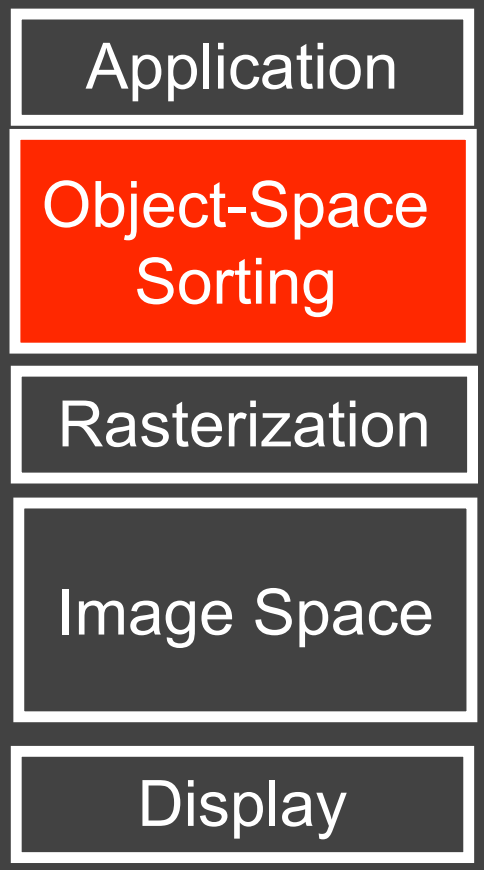


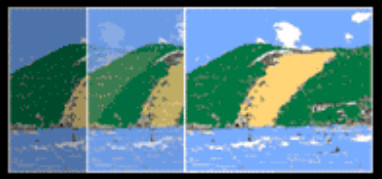
SIBGRAPI 2005

Overview

- Recent advances in GPU programmability
- *k*-Buffer: A fragment stream sorter
- Hardware-Assisted Visibility Sorting
- Dynamic Level-of-Detail

Sorting

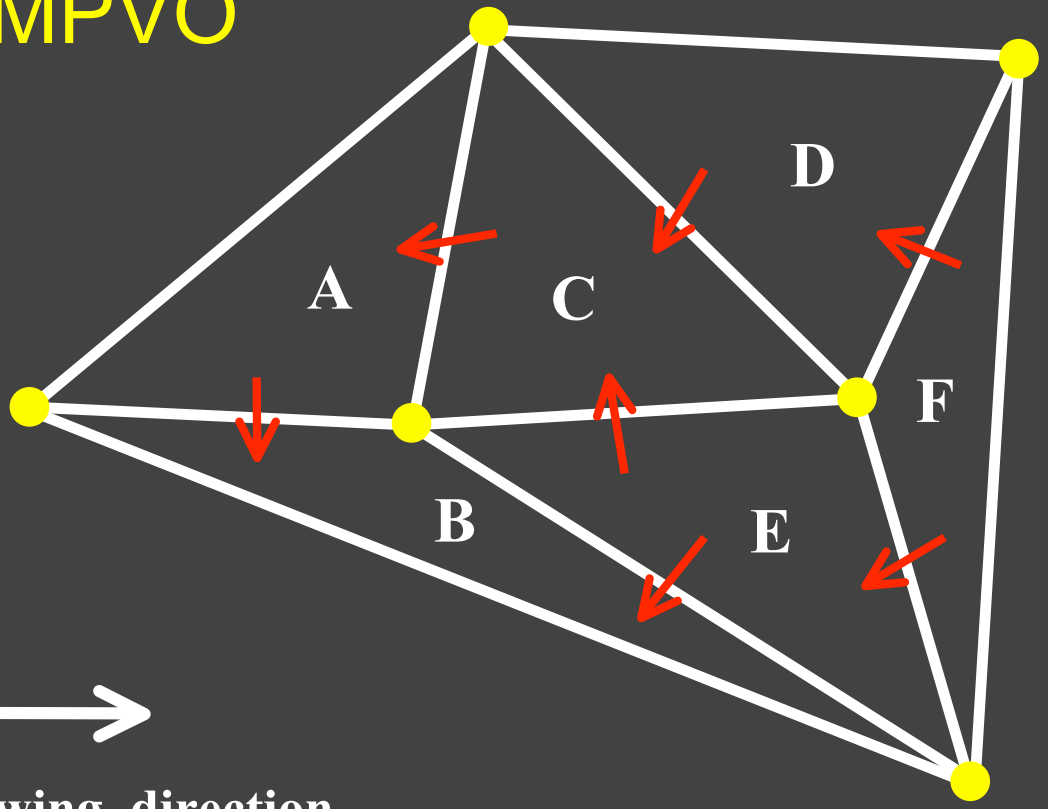

Application
Object-Space Sorting
Rasterization
Image Space
Display



SIBGRAPI 2005

Object-Space Sorting

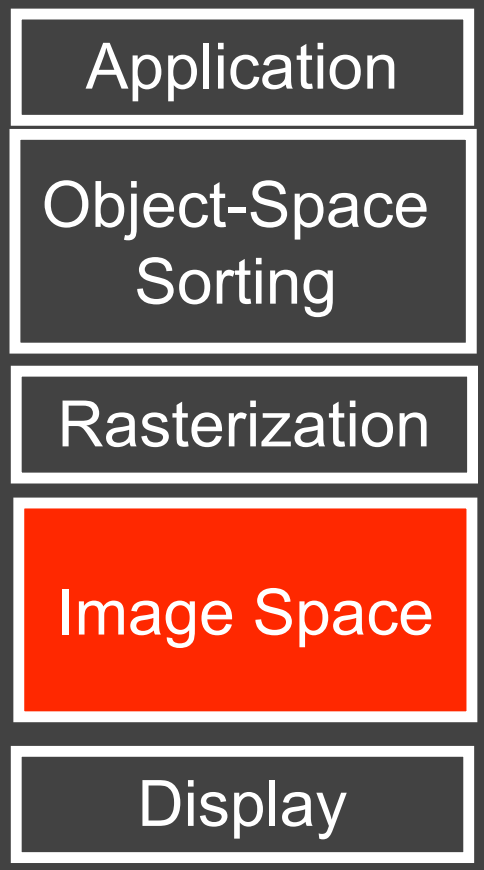
MPVO

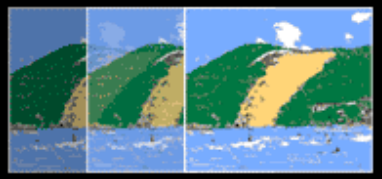


B < A
A < C
B < E
C < D
C < E
E < F
D < F

[Williams]

Sorting

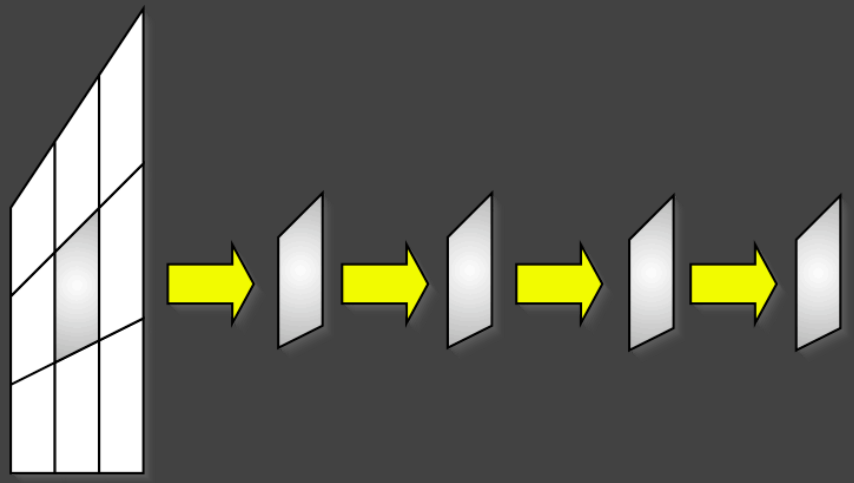

Application
Object-Space Sorting
Rasterization
Image Space
Display

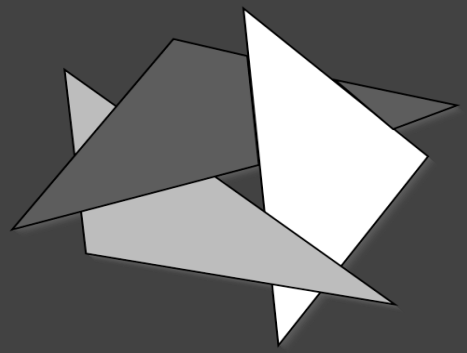


SIBGRAPI 2005

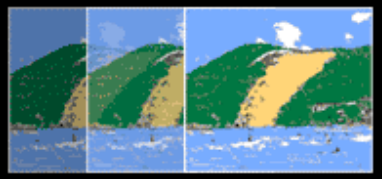
Image-Space Sorting

A-Buffer

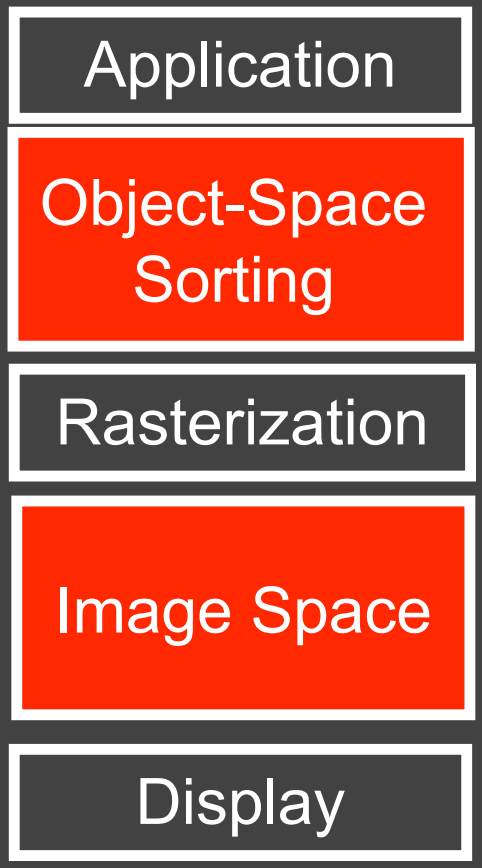


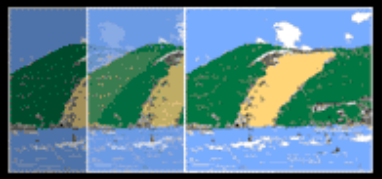


[Carpenter]



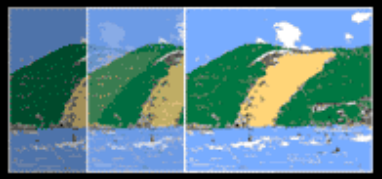
Sorting


Application
Object-Space Sorting
Rasterization
Image Space
Display



k-Buffer

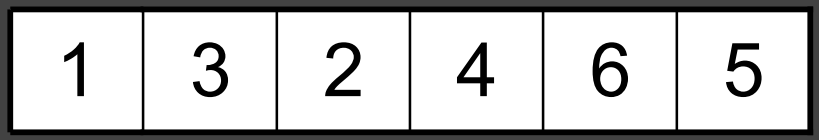
- Fixed-size A-Buffer
- As a new pixel is inserted, another is removed
- Can efficiently sort a *k*-Nearly Sorted Sequence (*k*-NSS)



SIBGRAPI 2005

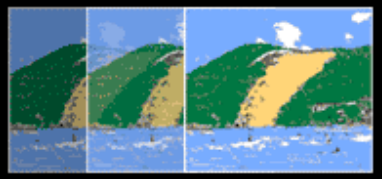
k-Buffer

input



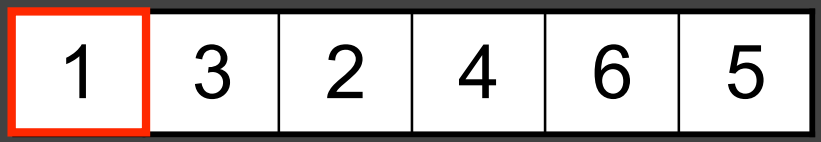
k-buffer

output



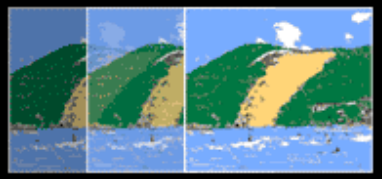
k-Buffer

input

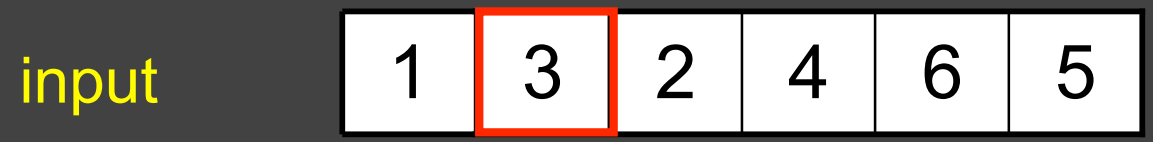


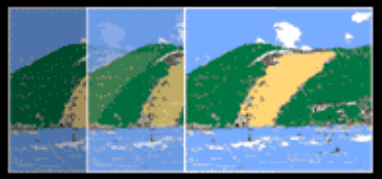
k-buffer

output

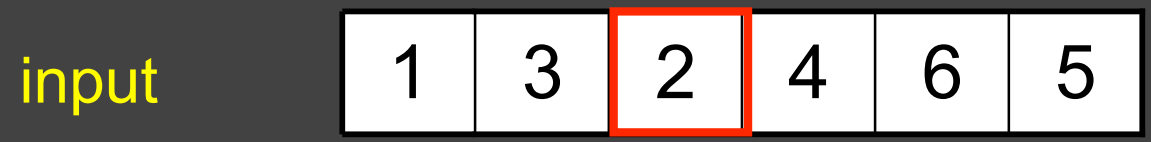


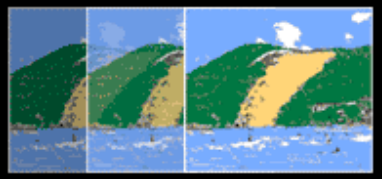
k-Buffer





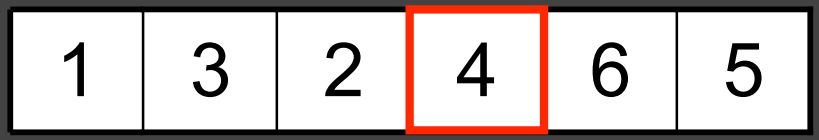
k-Buffer



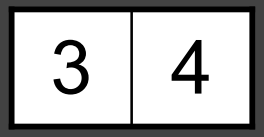


k-Buffer

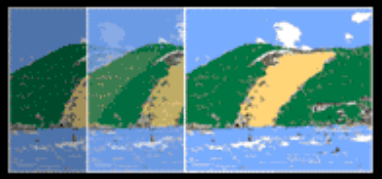
input



k-buffer



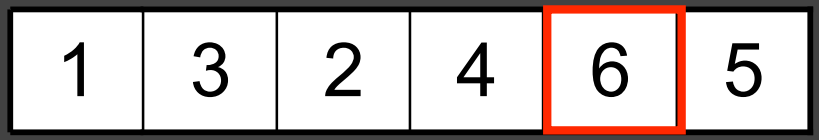
output



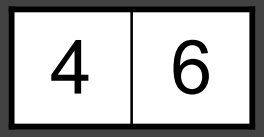
SIBGRAPI 2005

k-Buffer

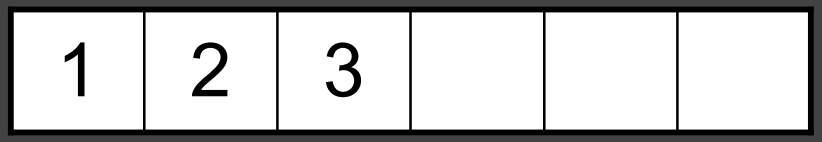
input

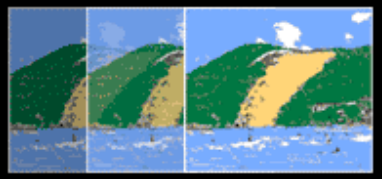


k-buffer



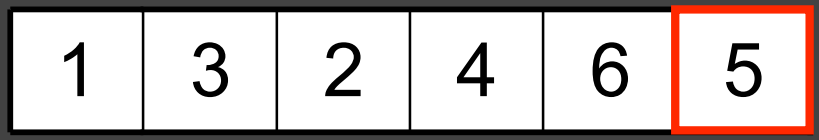
output



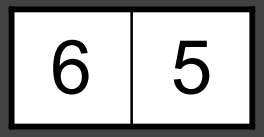


k-Buffer

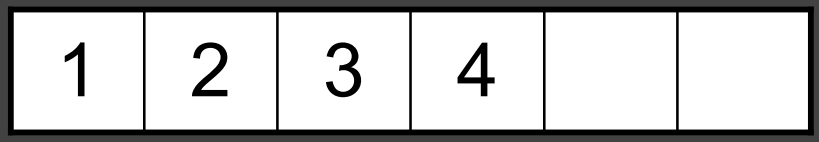
input

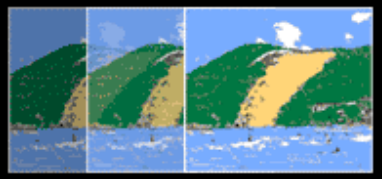


k-buffer



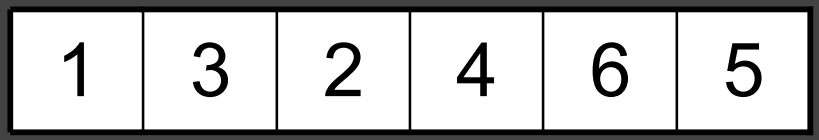
output



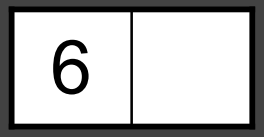


k-Buffer

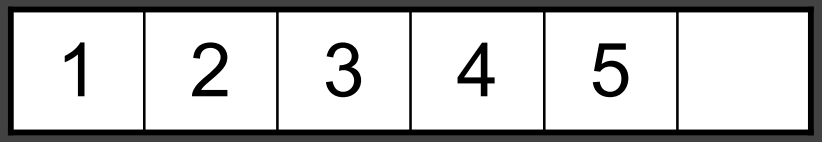
input

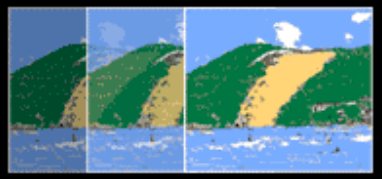


k-buffer



output

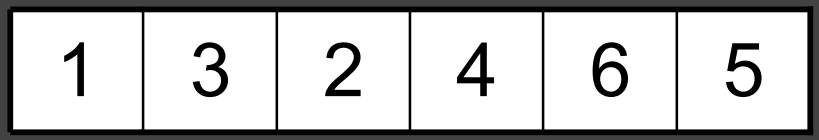




SIBGRAPI 2005

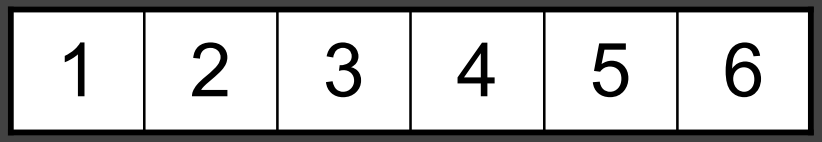
k-Buffer

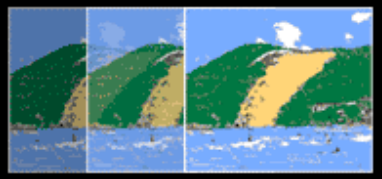
input



k-buffer

output

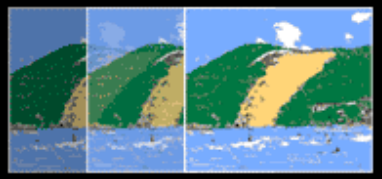




SIBGRAPI 2005

Overview

- Recent advances in GPU programmability
- *k*-Buffer: A fragment stream sorter
- Hardware-Assisted Visibility Sorting
- Dynamic Level-of-Detail



SIBGRAPI 2005

Object-Space Sorting

- Performed on CPU
- Sort faces by center
- Least Significant Digit Radix Sort
- Handles floating-point numbers

```
inline unsigned int float2fint (unsigned int f)
{
        return f ^ ((-(f >> 31)) | 0x80000000);
}
```

- Results: 15 million faces/sec

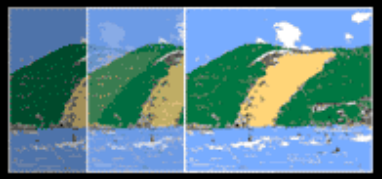
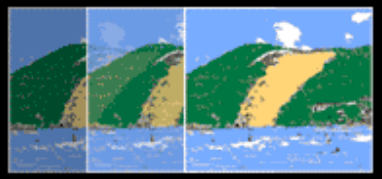


Image-Space Sorting

- Performed on GPU
- Uses *k*-Buffer as a fragment stream sorter
- Keeps *k* entries per pixel, each entry contains a fragment's scalar value and distance from the viewpoint (*v,d*)
- An incoming fragment replaces the entry that is closest to the eye (front-to-back compositing)

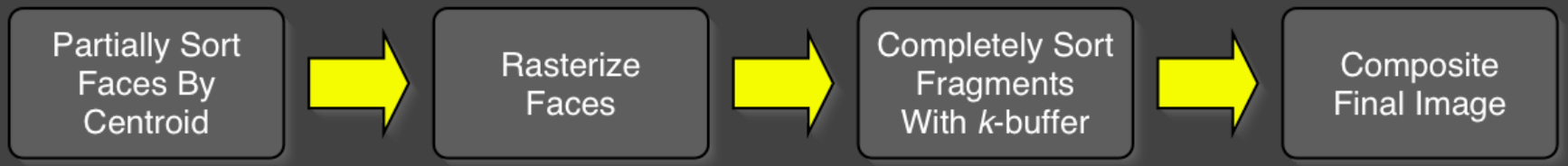


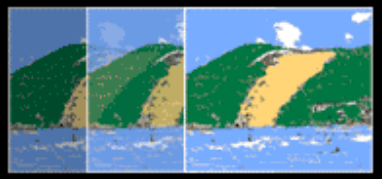
SIBGRAPI 2005

Hardware-Assisted Visibility Sorting

Sort in image-space and object-space

1. Approximate sort in object-space
2. Complete sort in image-space



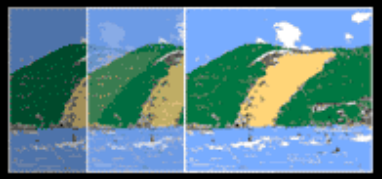


SIBGRAPI 2005

k-Buffer In Hardware

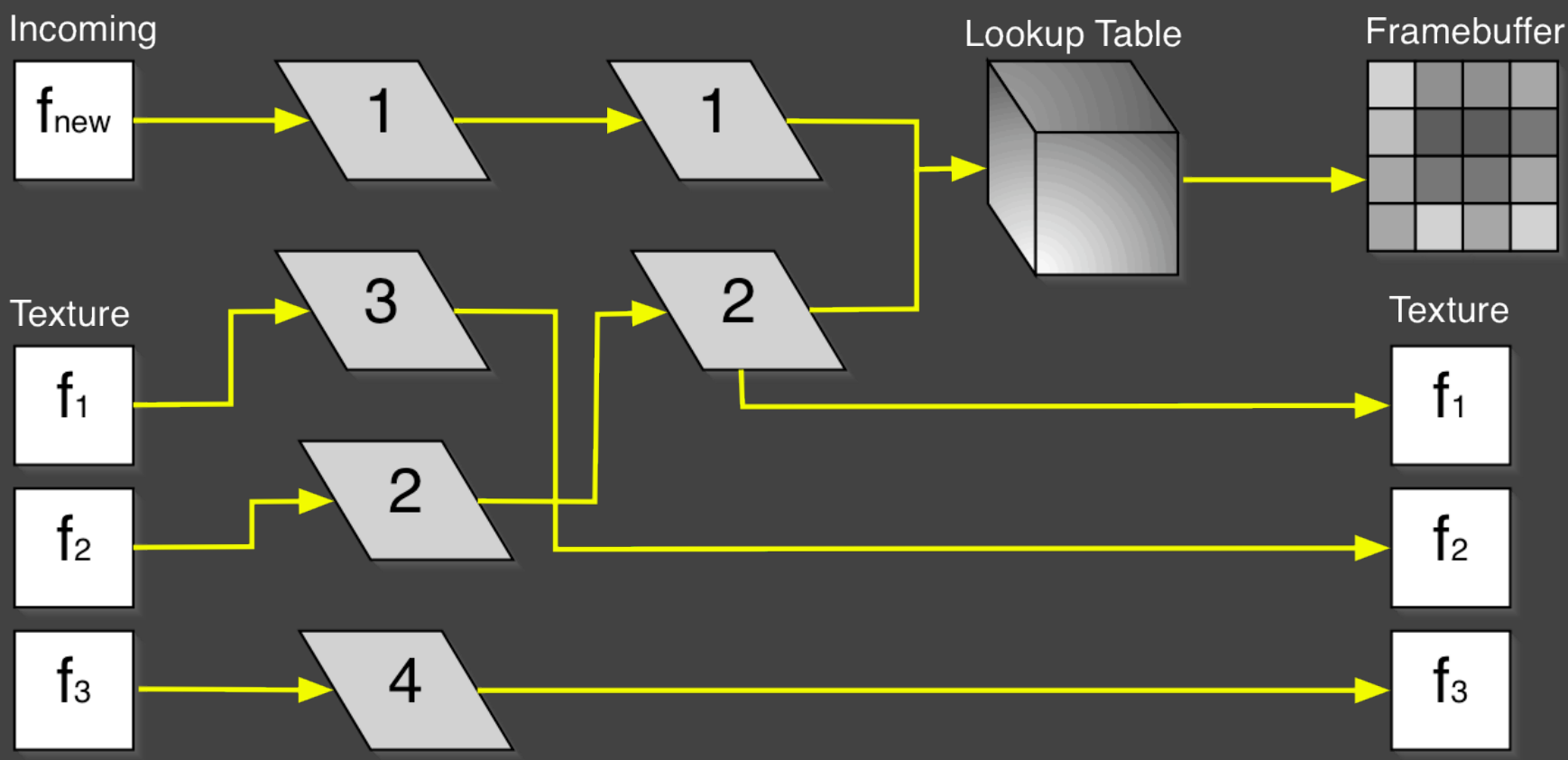
- Use MRTs
- Attach 4 32-bit floating-point RGBA textures to FBO as color attachments

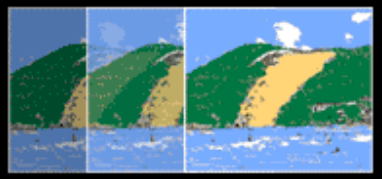
Texture 1	r_{comp}	g_{comp}	b_{comp}	a_{comp}
Texture 2	v_1	d_1	v_2	d_2
Texture 3	v_3	d_3	v_4	d_4
Texture 4	v_5	d_5	v_6	d_6



SIBGRAPI 2005

k-Buffer in Hardware

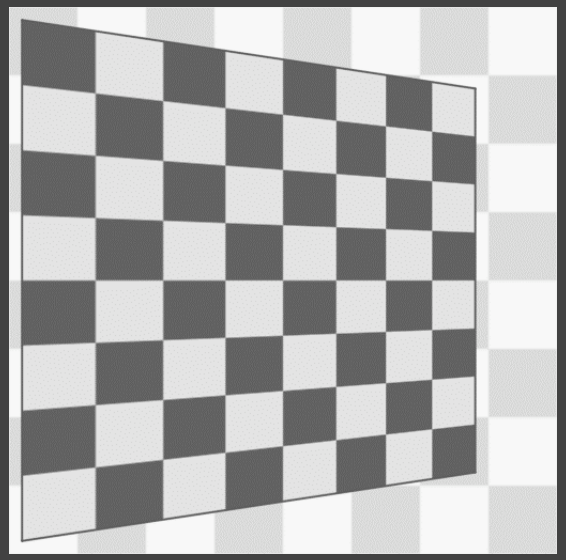




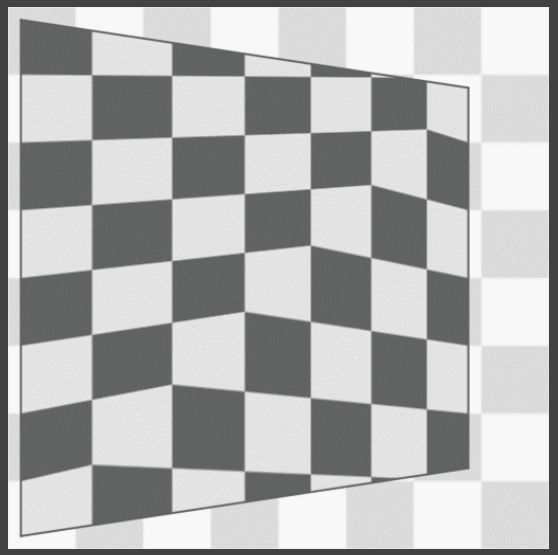
SIBGRAPI 2005

Details

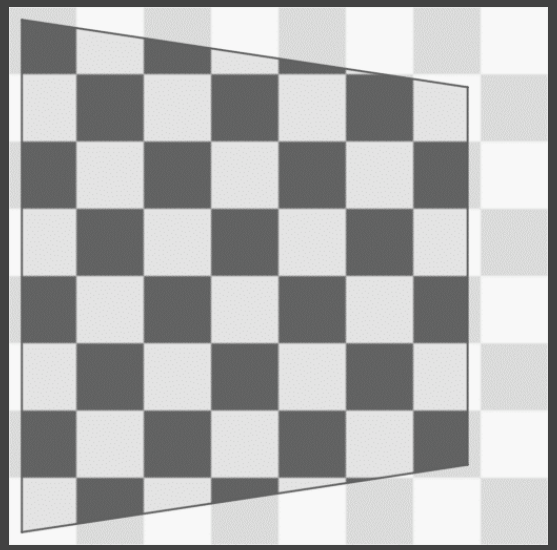
- Fix incorrect texture coordinates caused by perspective-correct interpolation



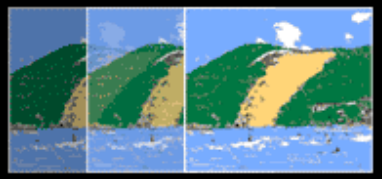
Perspective Correct



Projecting vertices to find tex coords

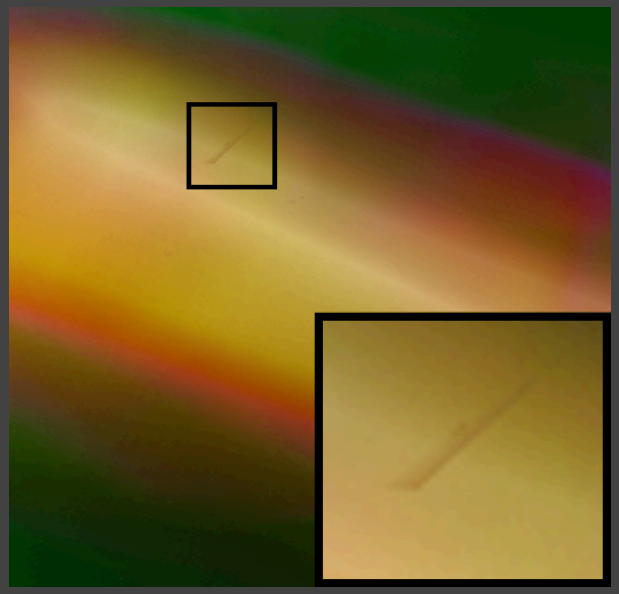


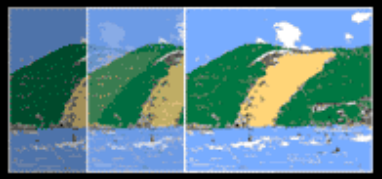
Projecting tex coords in shader



Details

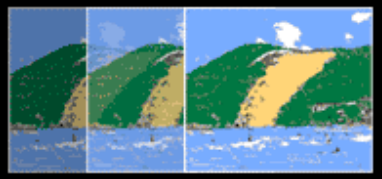
- Simultaneously reading and writing to a texture is undefined when fragments are rasterized in parallel





Details

- Initialization and Termination
- Non-convex objects



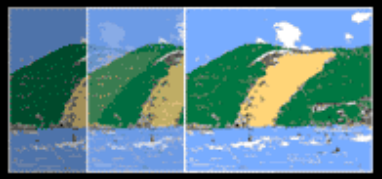
Experiments

Environment

- 3.2 GHz Pentium 4
- 2048 MB RAM
- Windows XP
- ATI Radeon 9800 Pro

Results

- *k*-Buffer analysis
- Performance results



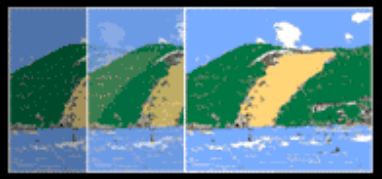
SIBGRAPI 2005

k-Buffer Analysis

Accuracy Analysis

- *k* depth required to render datasets
- Max values from 14 fixed viewpoints

Dataset	Max A	Max *k*	*k* > 2	*k* > 6
Spx2	476	22	10,262	512
Torso	649	15	43,317	1,683
Fighter	904	3	1	0

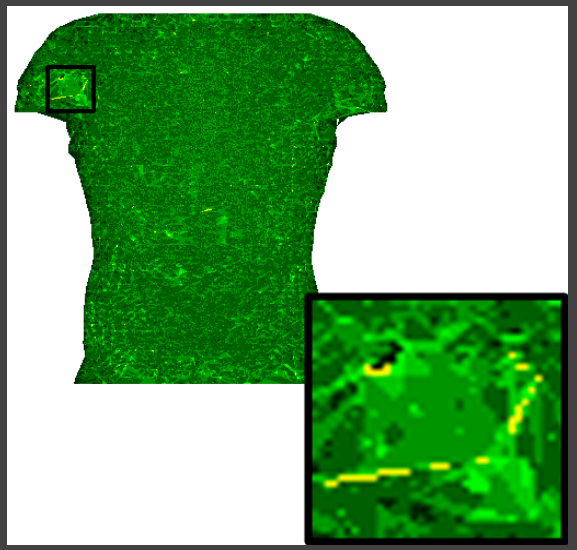


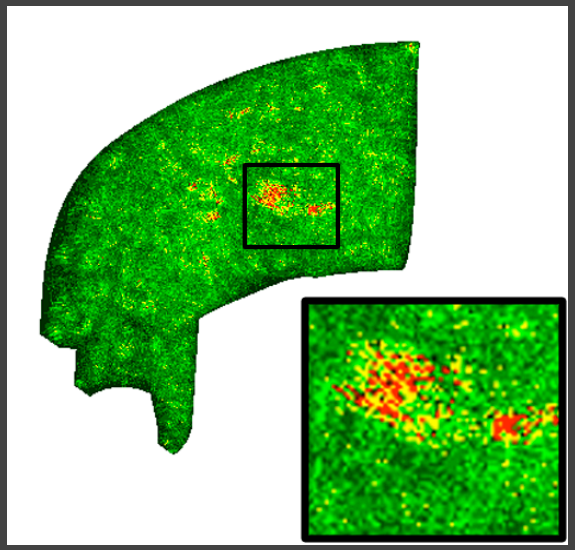
SIBGRAPI 2005

k-Buffer Analysis

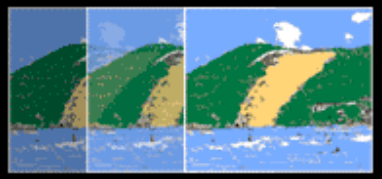
Distribution Analysis

- Shows the actual pixels that require large *k* depths to render correctly





k ≤ 2: green
2 < *k* ≤ 6: yellow
k > 6: red



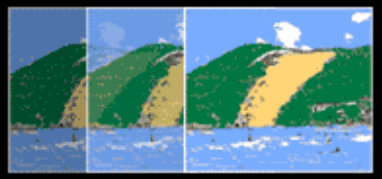
Results

Performance

- 512^2 viewport with a 128^3 pre-integrated lookup table

GPU Sorting:

Dataset	Cells	k = 2 fps	k = 2 tets/s	k = 6 fps	k = 6 tets/s
Spx2	0.8 M	2.07	1712 K	1.7	1407 K
Torso	1.1 M	3.13	3390 K	1.86	1977 K
Fighter	1.4 M	2.41	3387 K	1.56	2190 K



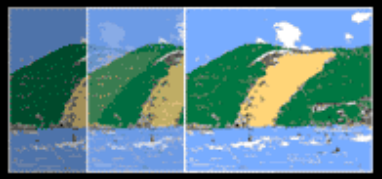
Results

Performance

- CPU sorting + GPU sorting and compositing
- Pipeline optimization = max(CPU, GPU)

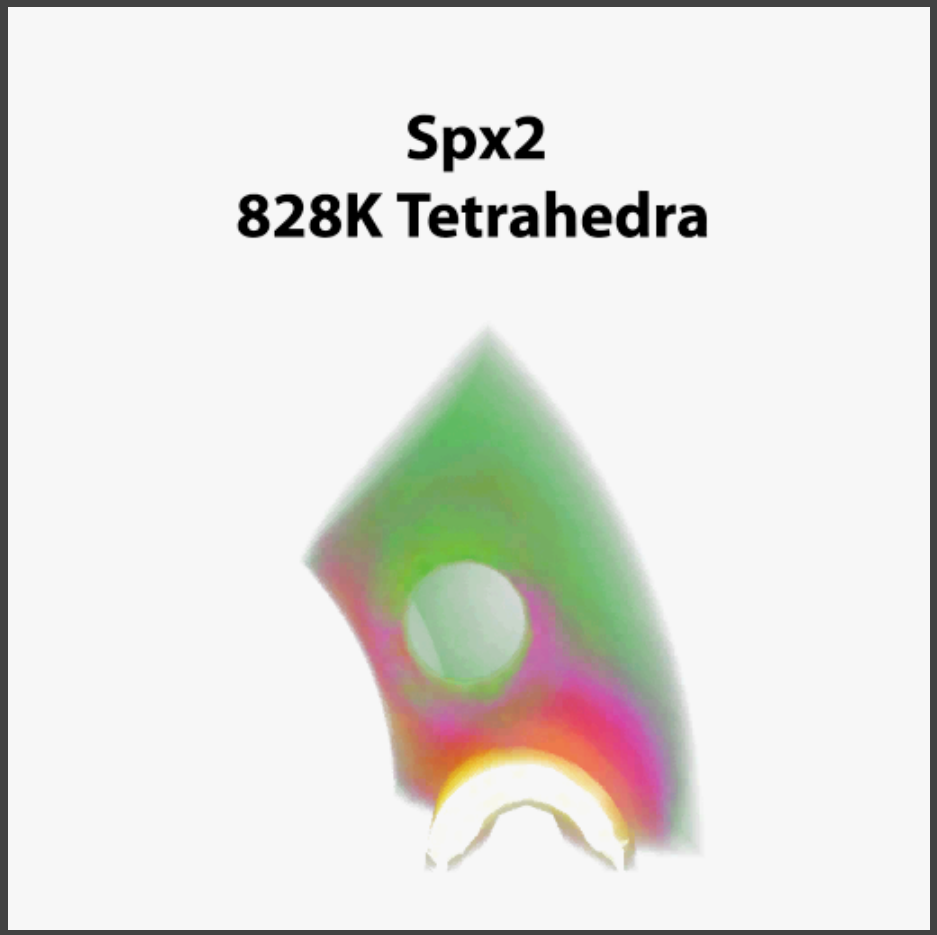
Total Time:

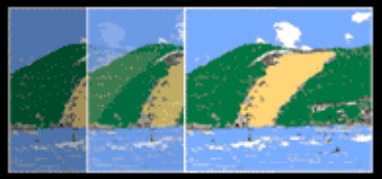
Dataset	Cells	CPU	GPU	Total	Tets/s
Spx2	0.8 M	160 ms	368 ms	528 ms	1568 K
Torso	1.1 M	210 ms	390 ms	600 ms	1805 K
Fighter	1.4 M	268 ms	505 ms	773 ms	1816 K



SIBGRAPI 2005

Movie

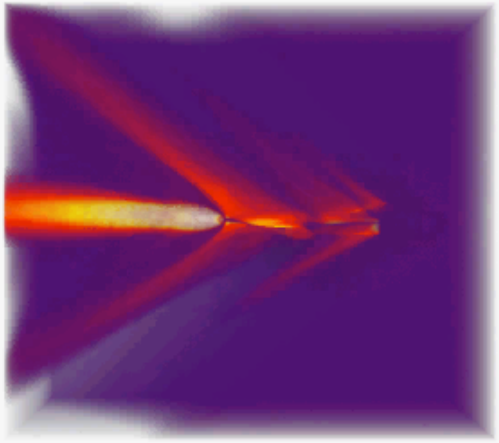


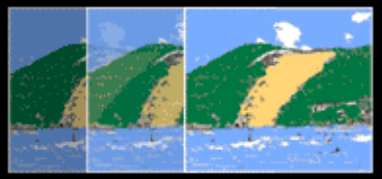


SIBGRAPI 2005

Movie

Fighter
1.40M Tetrahedra

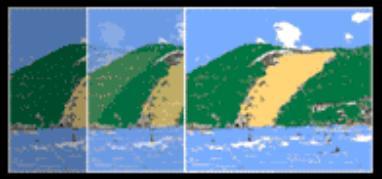




SIBGRAPI 2005

Conclusion

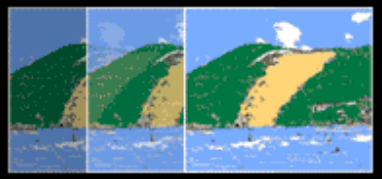
- Introduced the *k*-buffer and an efficient GPU implementation
- Fastest volume renderer for unstructured data
- Handles arbitrary non-convex meshes
- Requires minimal pre-processing of data
- Maximum data size is bounded by main memory
- Code is short and simple
- Can easily be extended



SIBGRAPI 2005

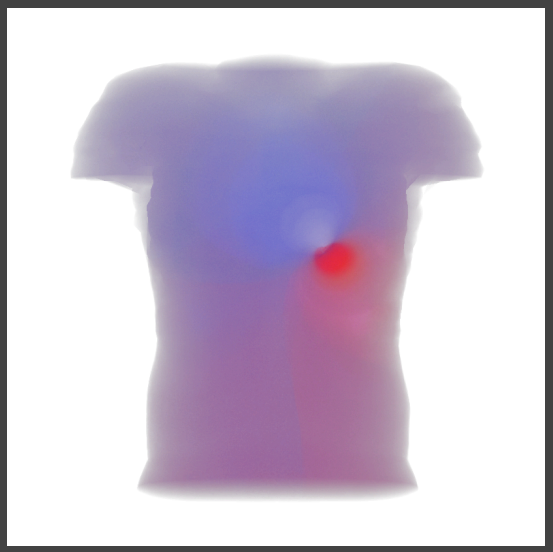
Overview

- Recent advances in GPU programmability
- *k*-Buffer: A fragment stream sorter
- Hardware-Assisted Visibility Sorting
- Dynamic Level-of-Detail

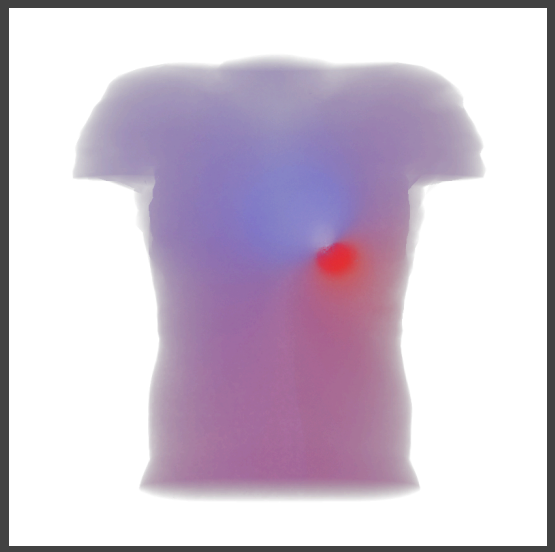


SIBGRAPI 2005

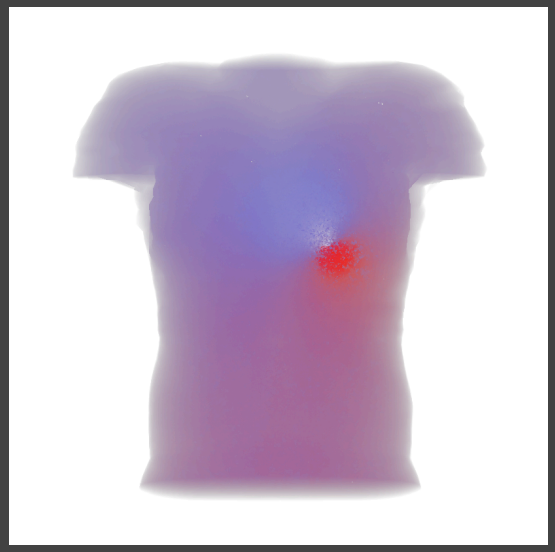
Dynamic Level-of-Detail



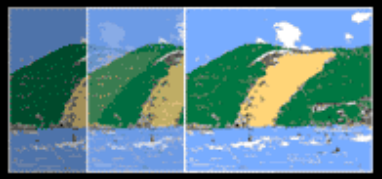
100%



25%



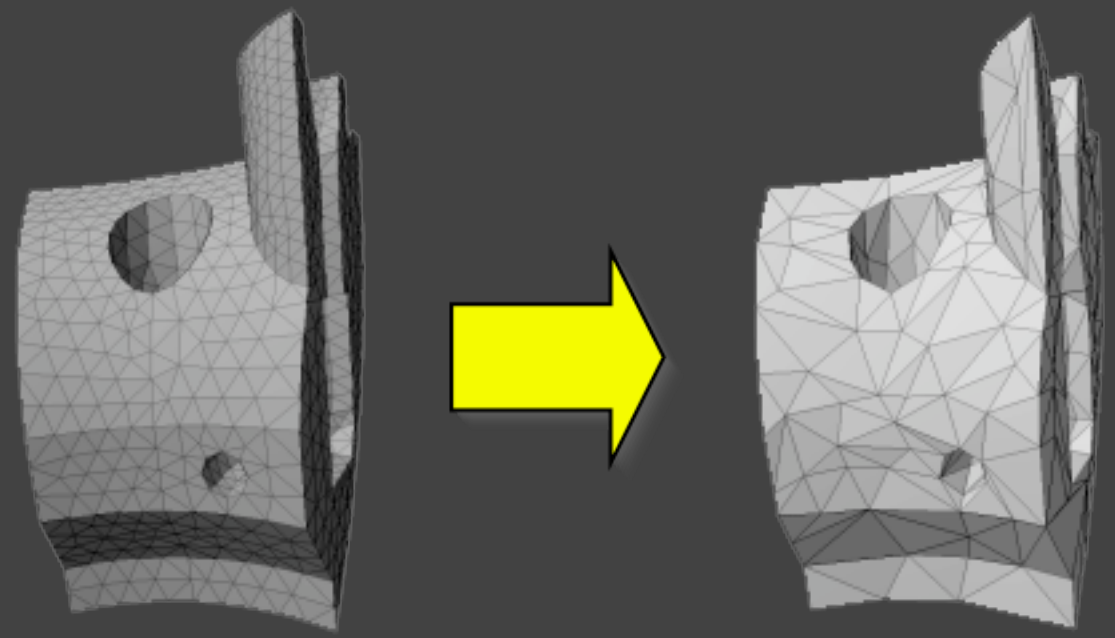
5%



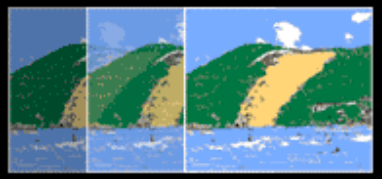
SIBGRAPI 2005

LOD Background

Geometric Approach



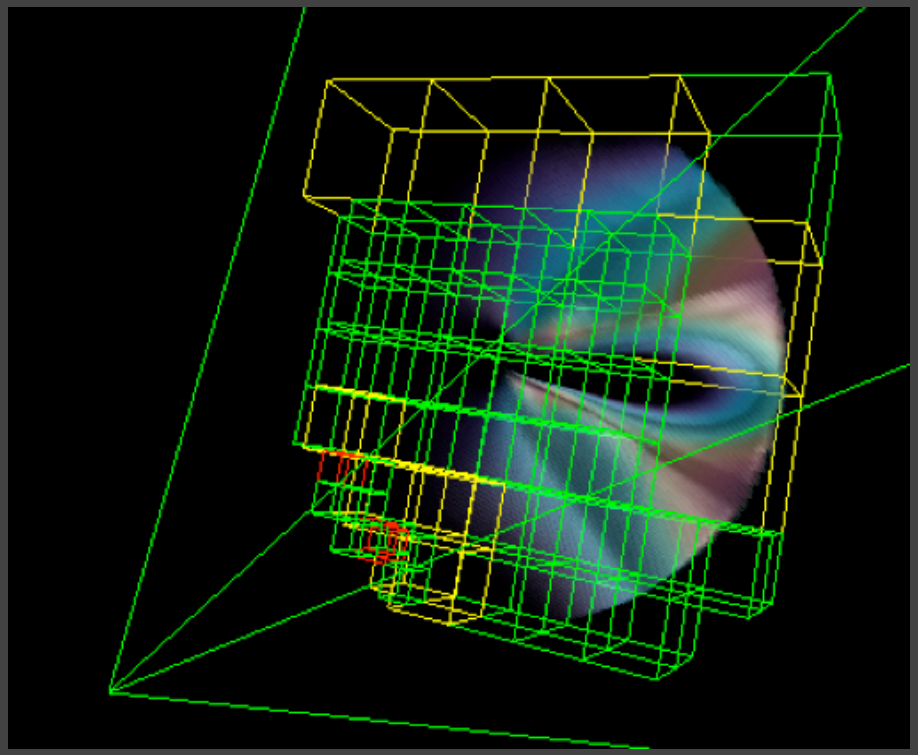
[Cignoni et al. 04]



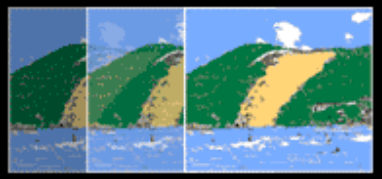
SIBGRAPI 2005

LOD Background

Texture Approach



[Leven et al. 02]



Definitions

Given a scalar field

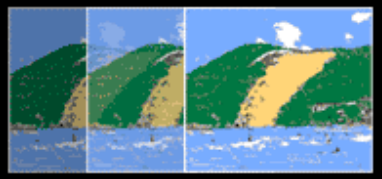
$$f : D \subseteq \Re^3 \rightarrow \Re$$

An approximation can be made such that

$$\left|\overline{f} - f\right| \leq \varepsilon \quad \text{and} \quad \left|\overline{D}\right| < \left|D\right|$$

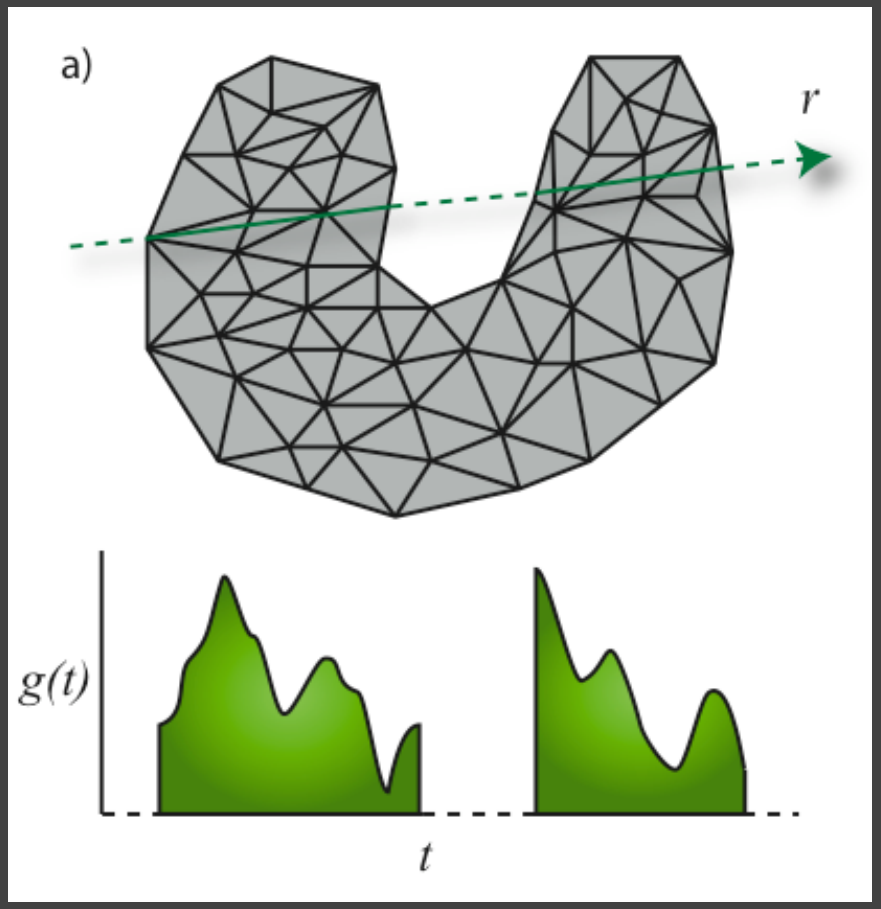
A ray passing through the domain forms a continuous function

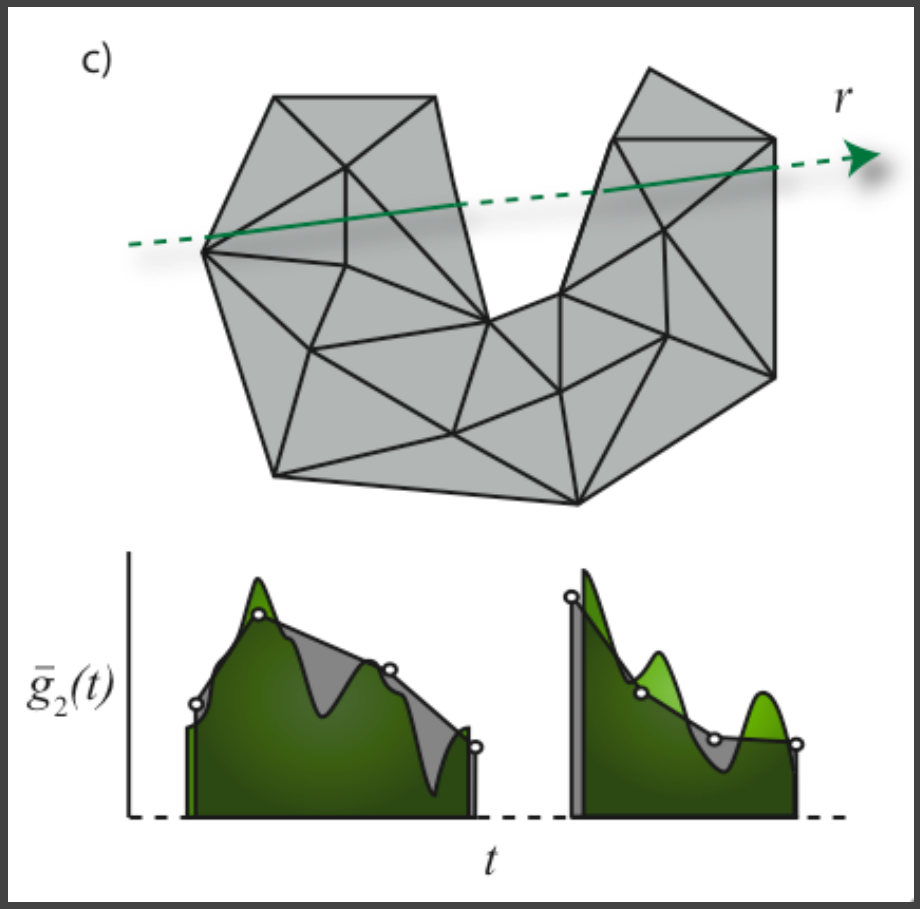
$$g(t) = f(r_0 + tr_d)$$

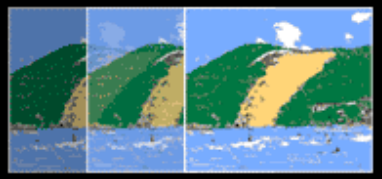


SIBGRAPI 2005

Domain-Based Simplification

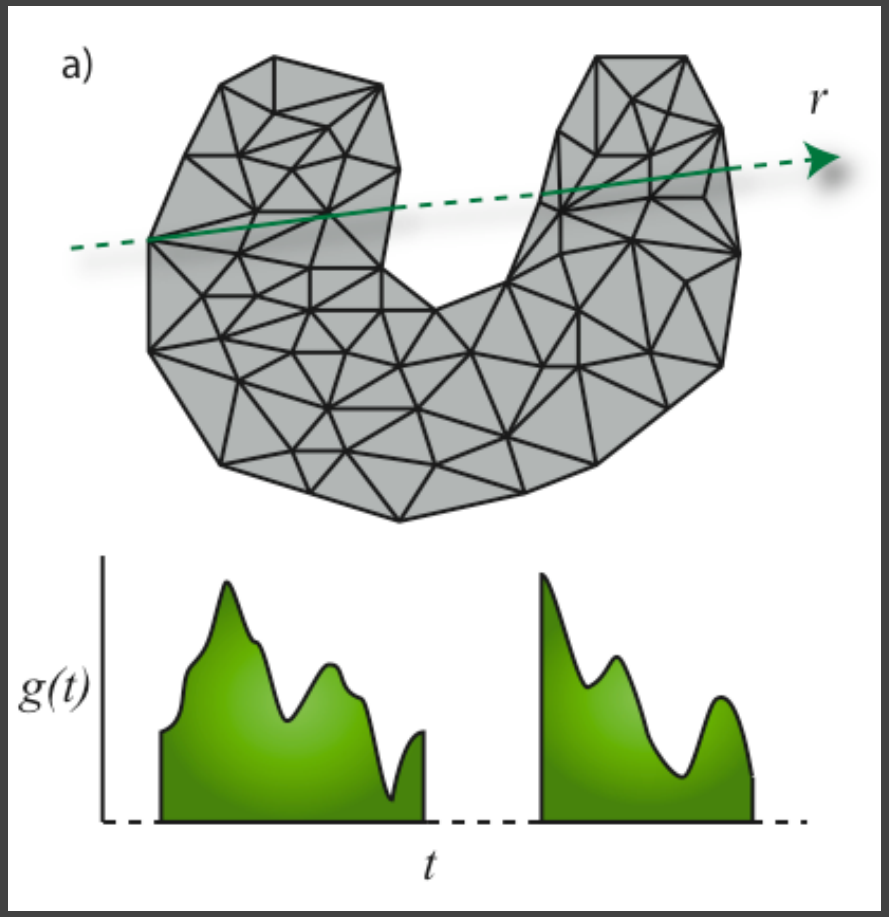


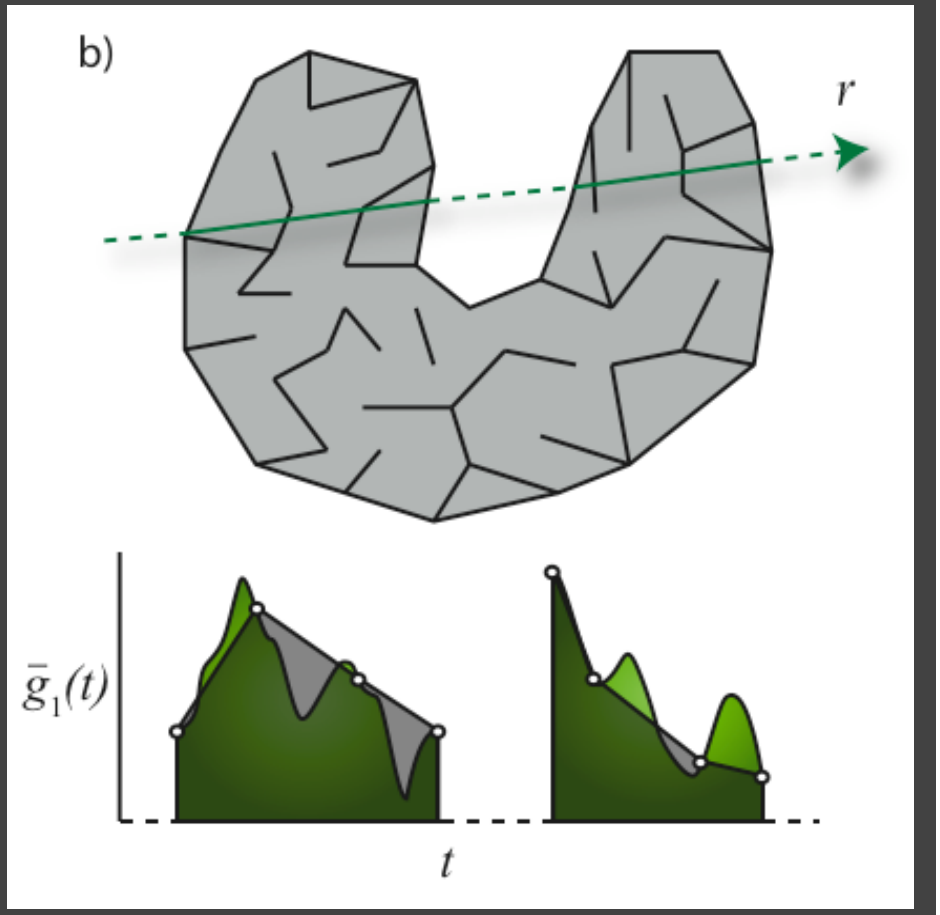


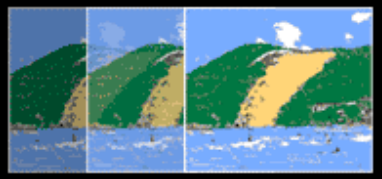


SIBGRAPI 2005

Sample-Based Simplification



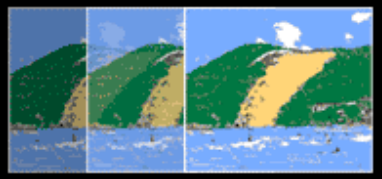




Domain vs. Sample

Domain-based simplification computes the exact volume integral over the approximate geometry

Sample-based simplification computes an approximate volume integral over the original geometry



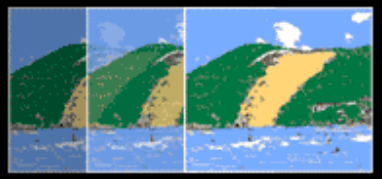
SIBGRAPI 2005

Dynamic Level-of-Detail

Face sub-sampling

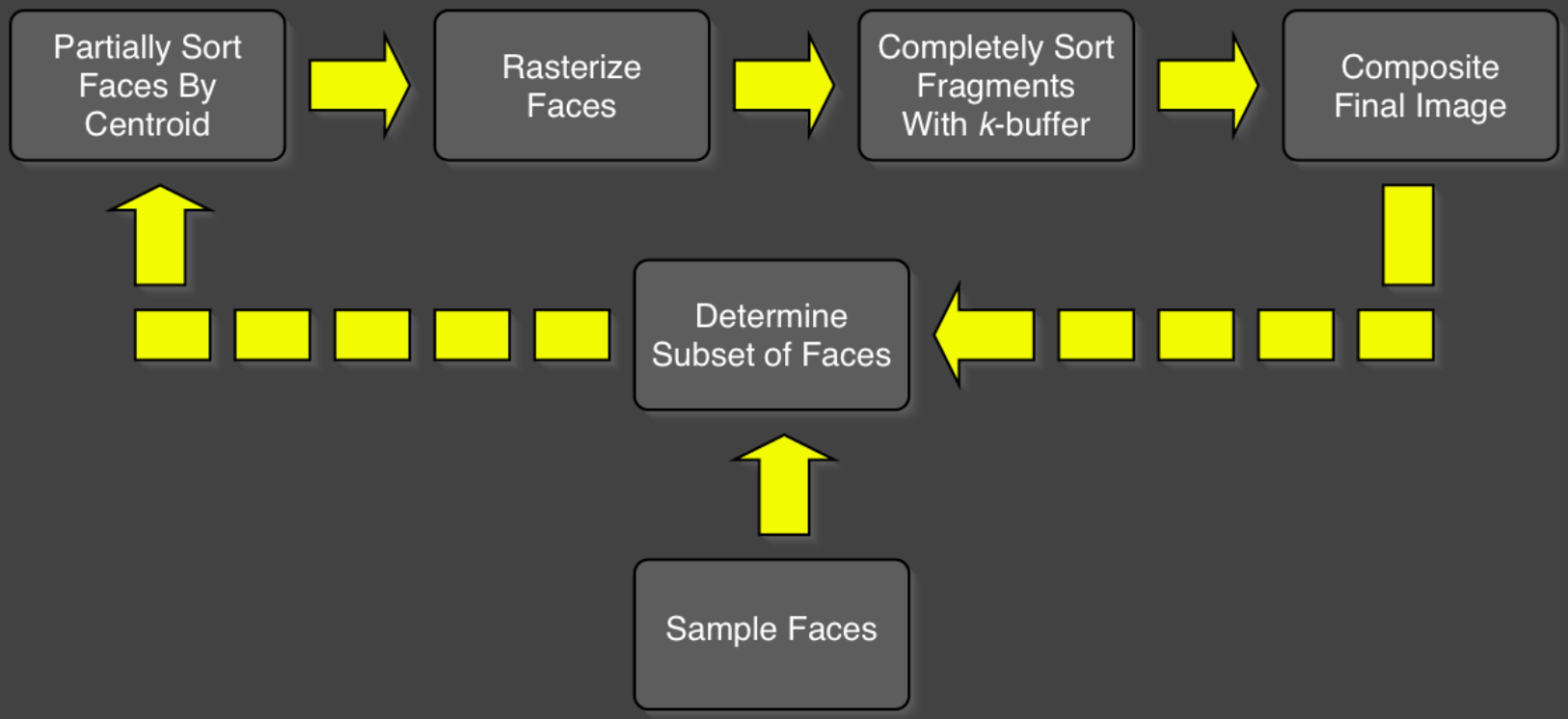
- Draw a subset of the original faces
- Base case: boundary faces
- Sample the internal faces

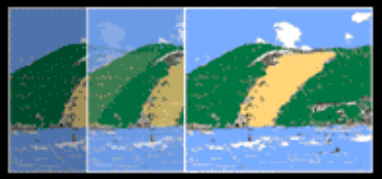
$$|I| = \frac{|I_{prev}| \times \text{TargetTime}}{\text{RenderTime}}$$



SIBGRAPI 2005

Dynamic Level-of-Detail

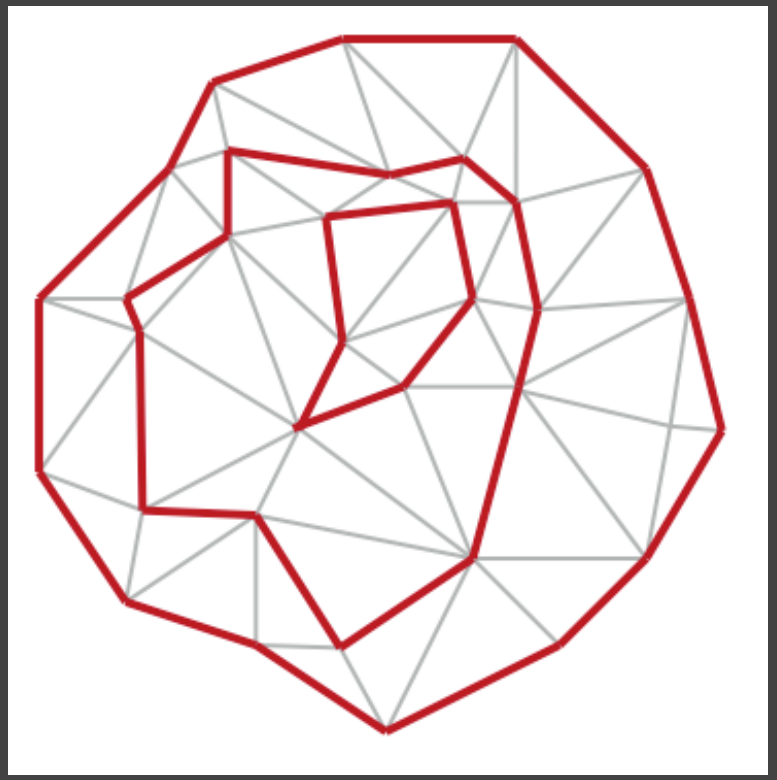


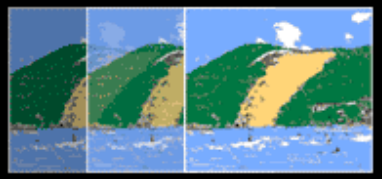


SIBGRAPI 2005

Sampling Strategies

Topology: target continuity

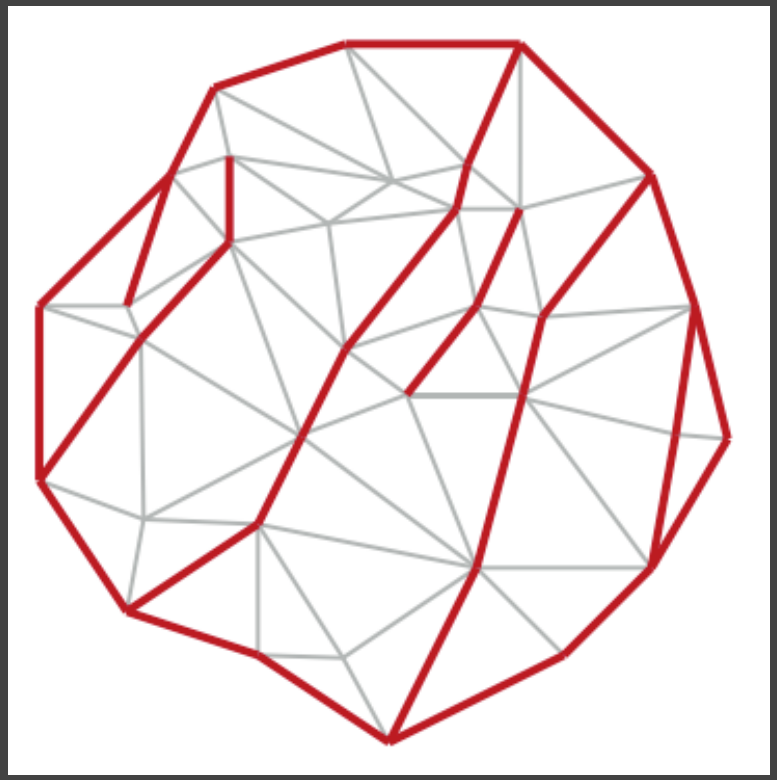


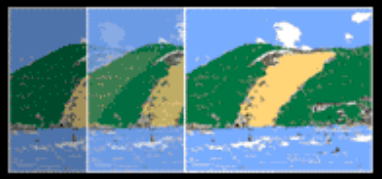


SIBGRAPI 2005

Sampling Strategies

View: target screen-space coverage

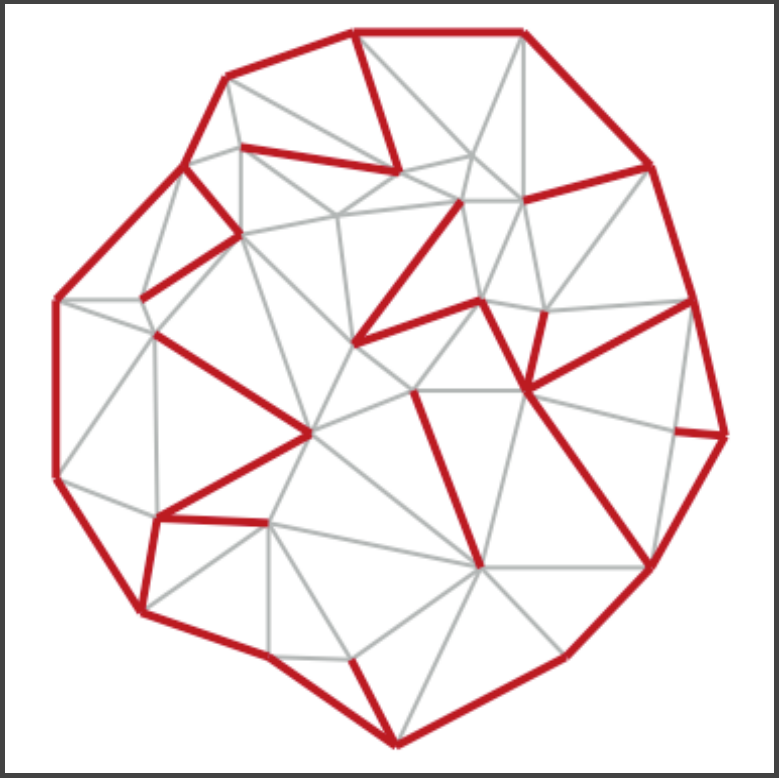


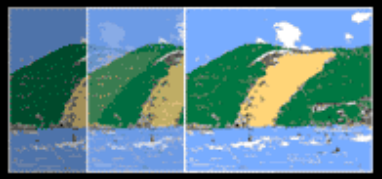


SIBGRAPI 2005

Sampling Strategies

Field: target histogram

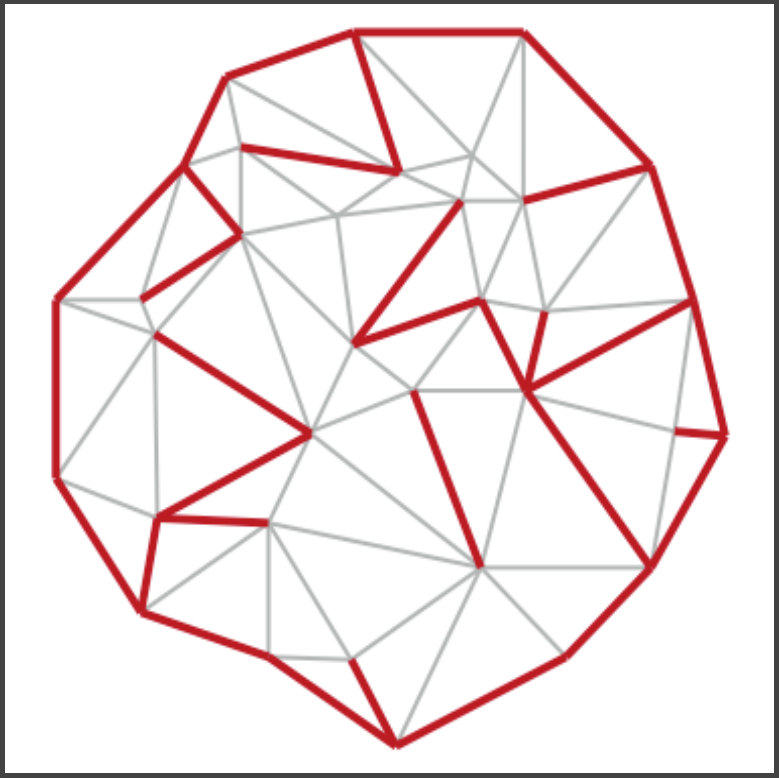


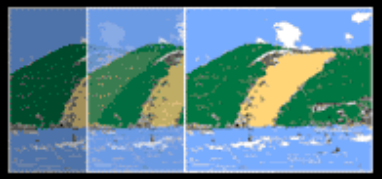


SIBGRAPI 2005

Sampling Strategies

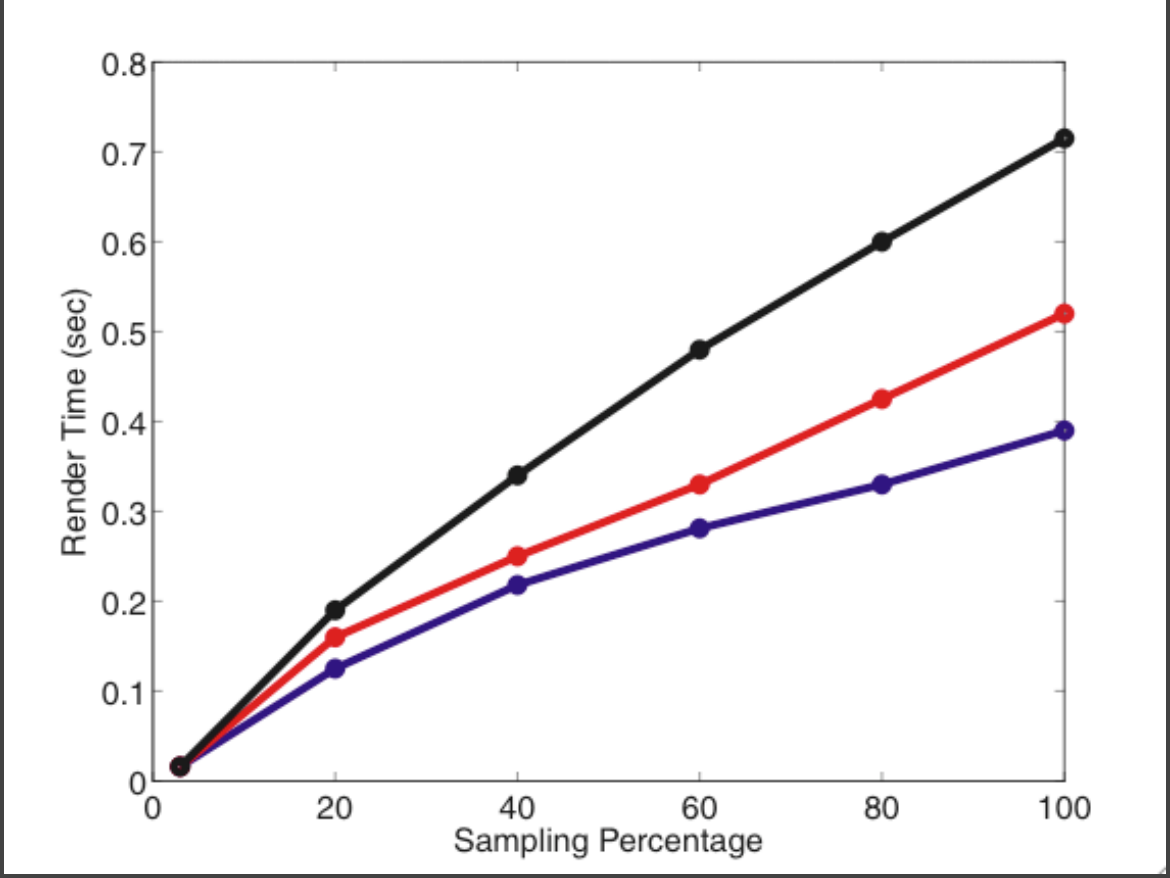
Area: target faces that cause greater error

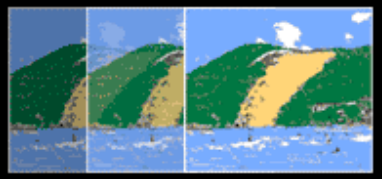




SIBGRAPI 2005

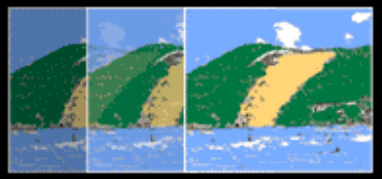
Results: Time





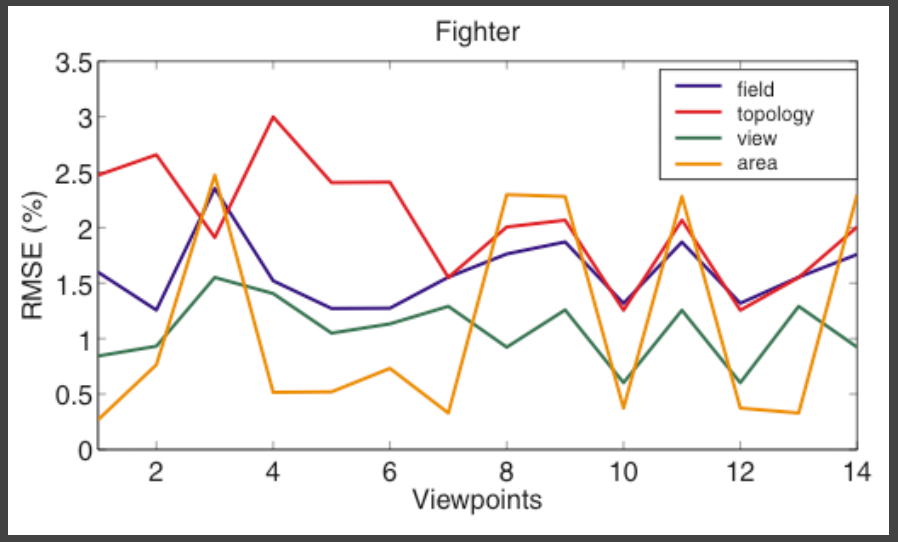
Results: Preprocessing

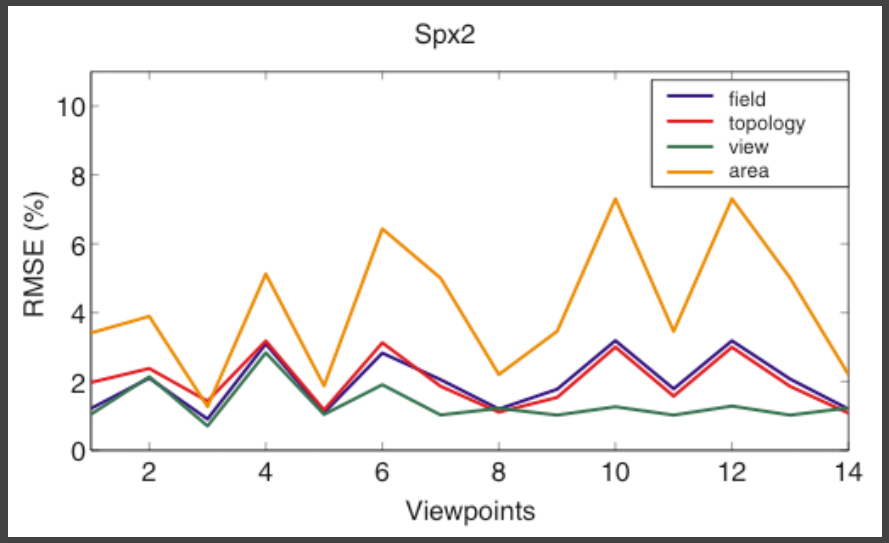
Dataset	Tets	Topology	View	Field	Area
Spx2	0.8 M	17.8	5.3	4.5	13.9
Torso	1.0 M	87.2	11.6	10.5	11.2
Fighter	1.4 M	75.6	15.3	13.9	15.3

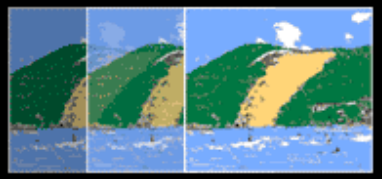


SIBGRAPI 2005

Results: Comparison

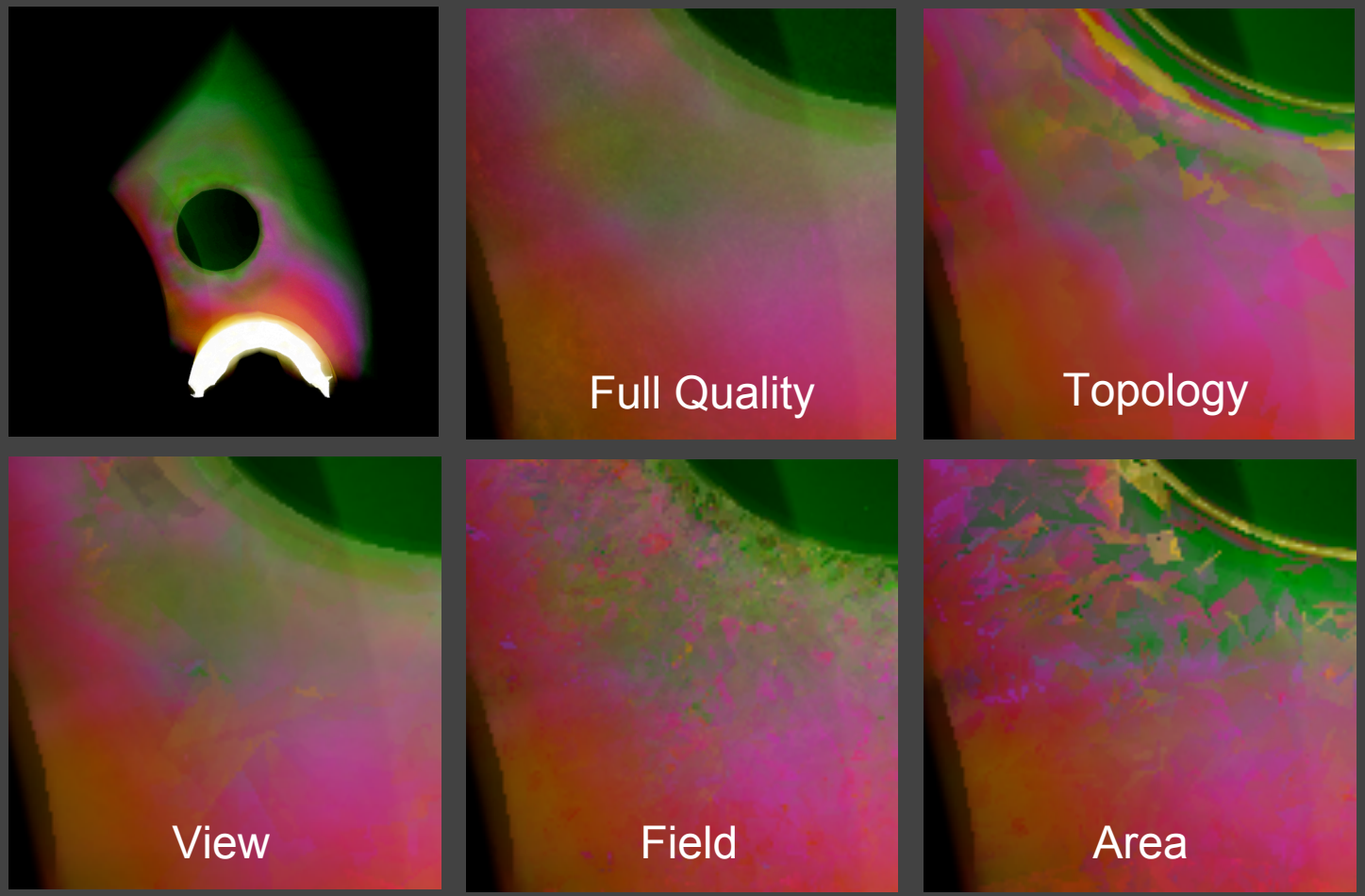


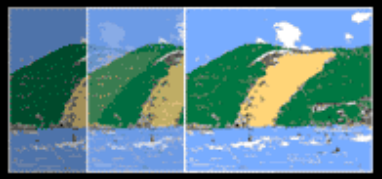




SIBGRAPI 2005

Results: Comparison





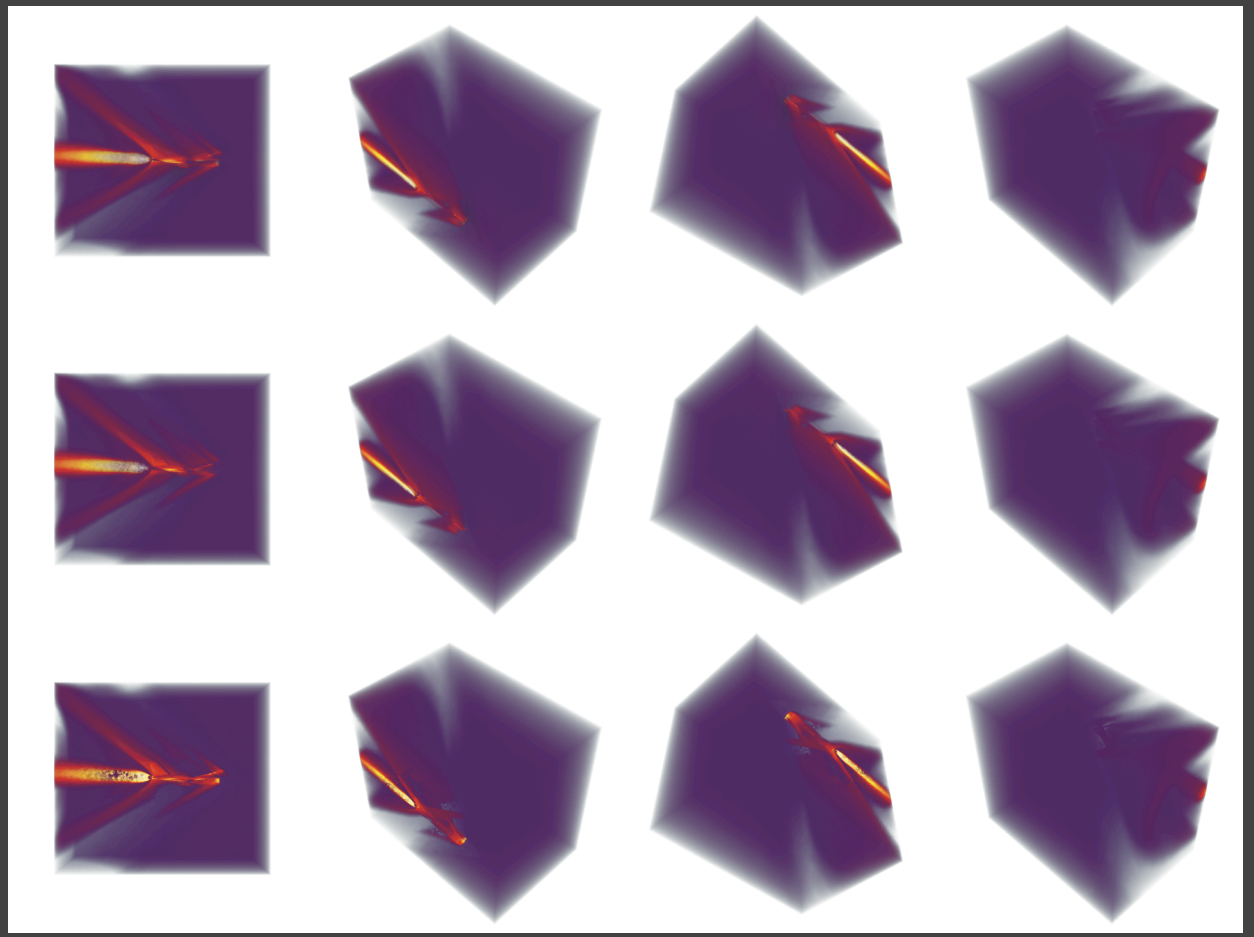
SIBGRAPI 2005

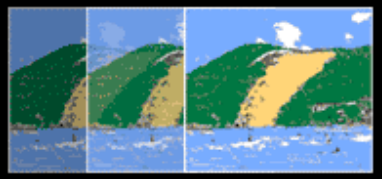
Results

100%
1.3 fps

15%
4.5 fps

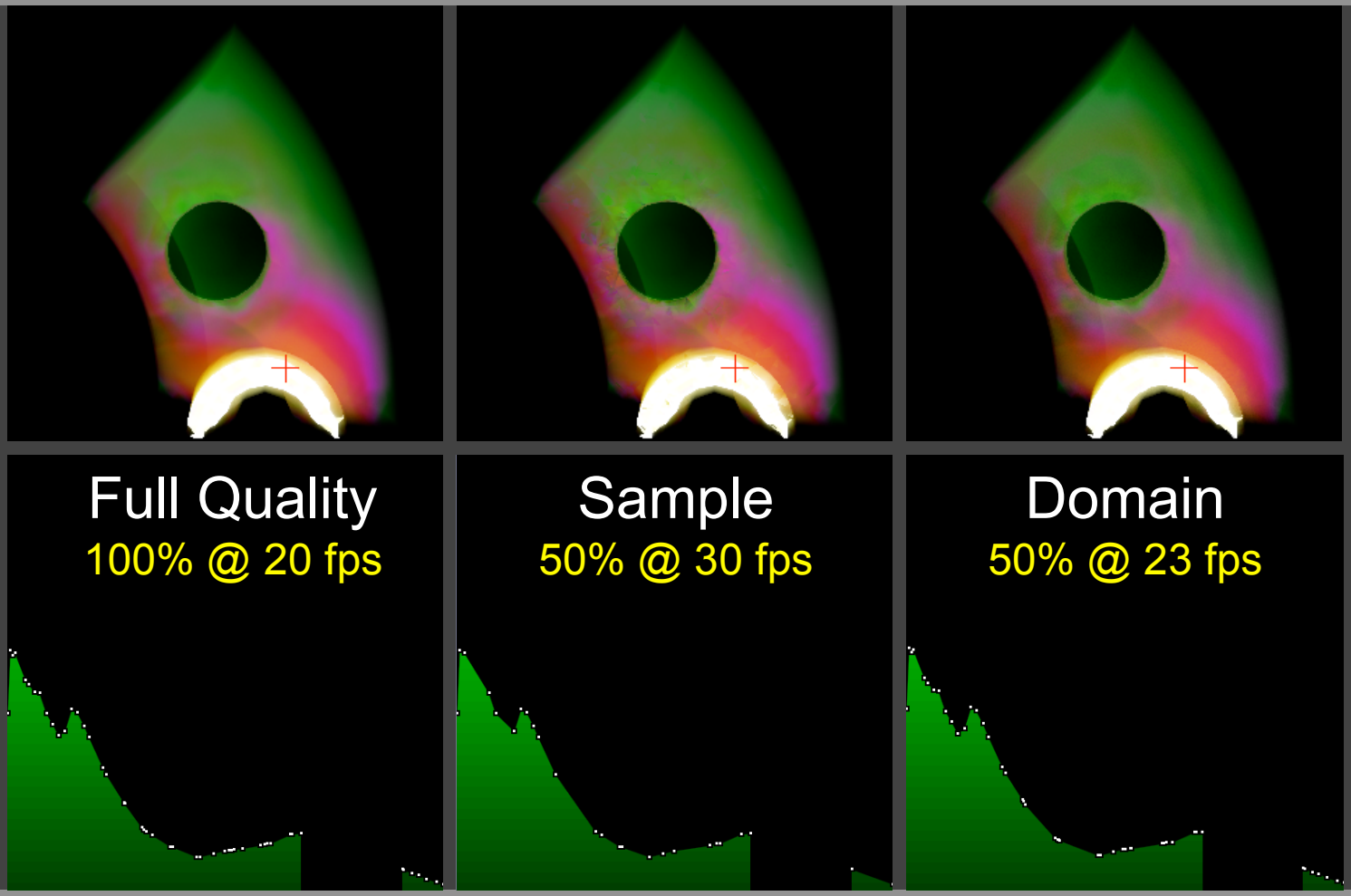
5%
10.0 fps

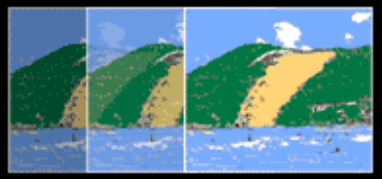




SIBGRAPI 2005

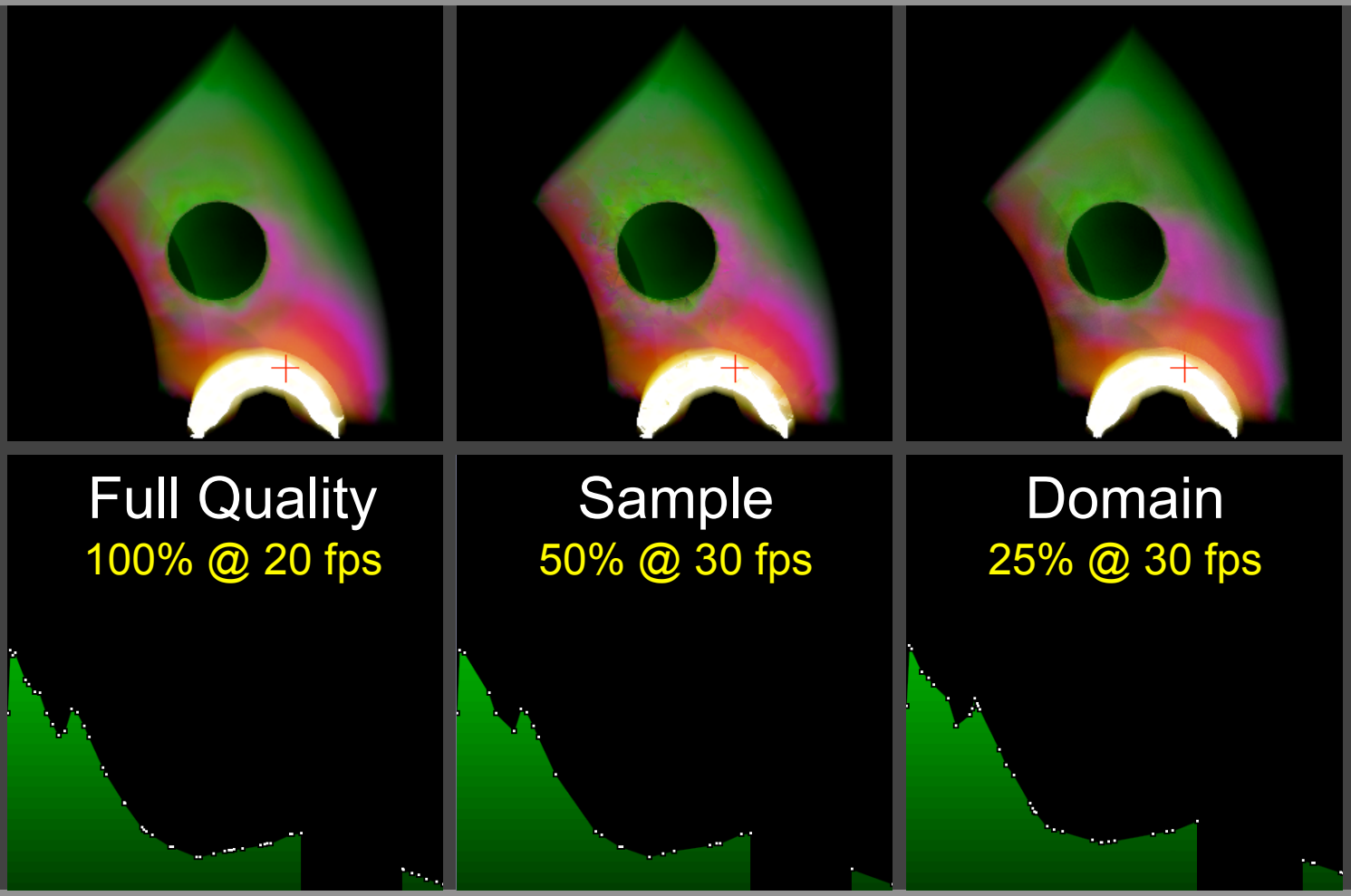
Comparison

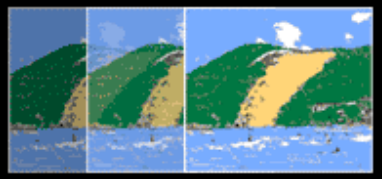




SIBGRAPI 2005

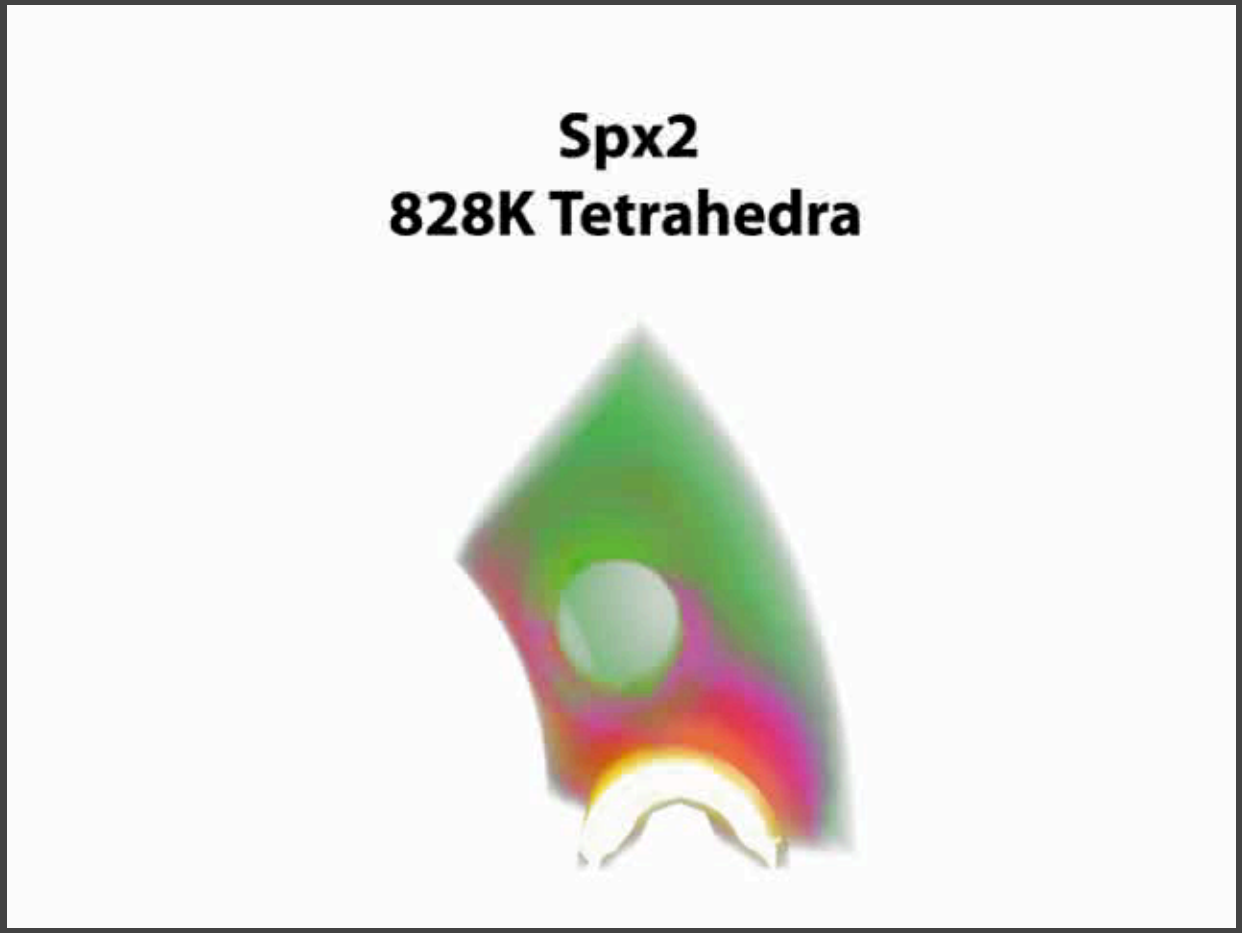
Comparison

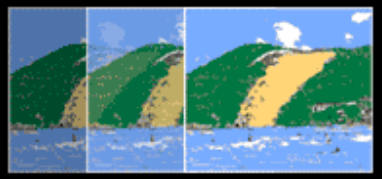




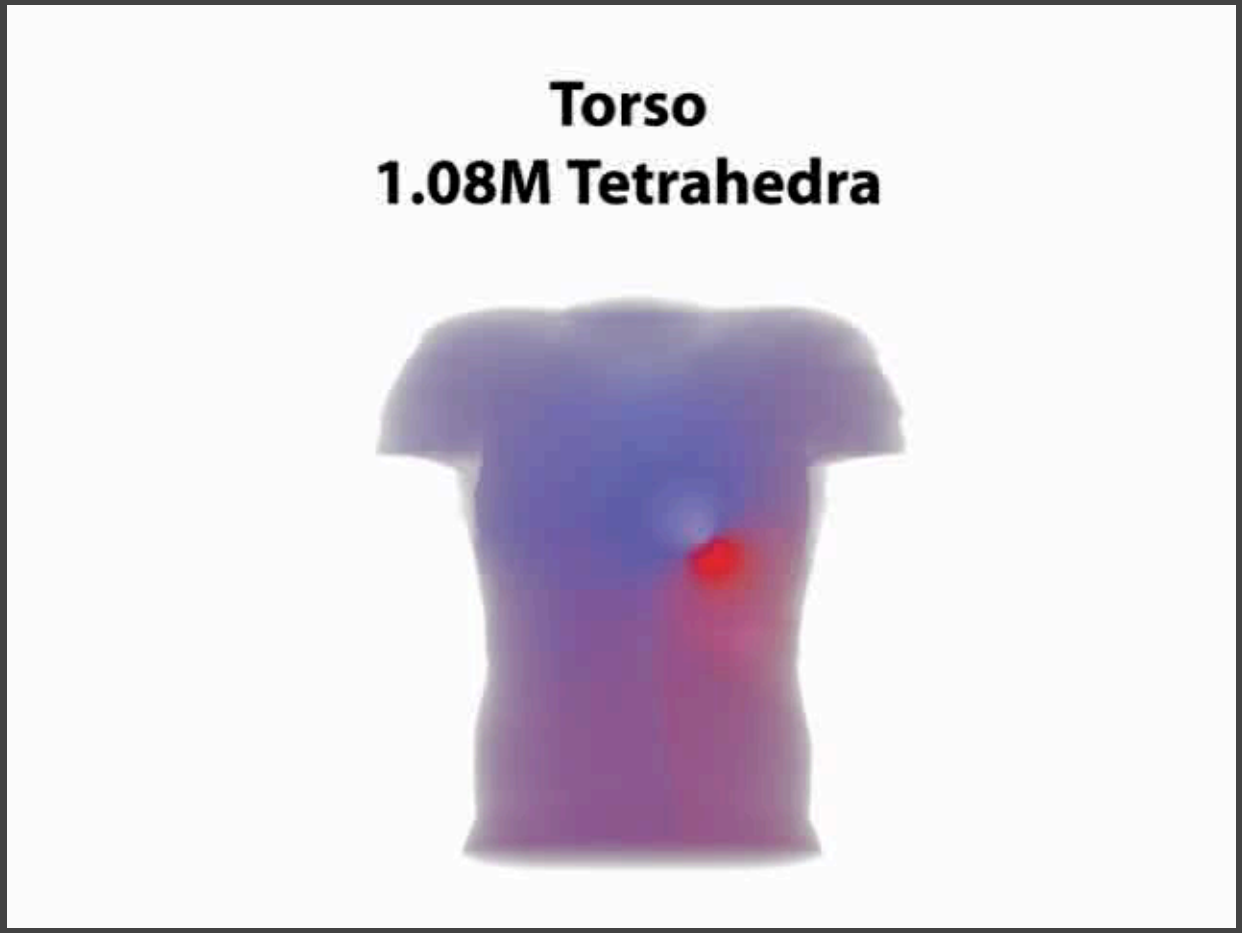
SIBGRAPI 2005

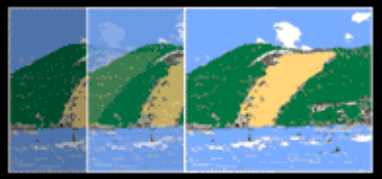
Movie





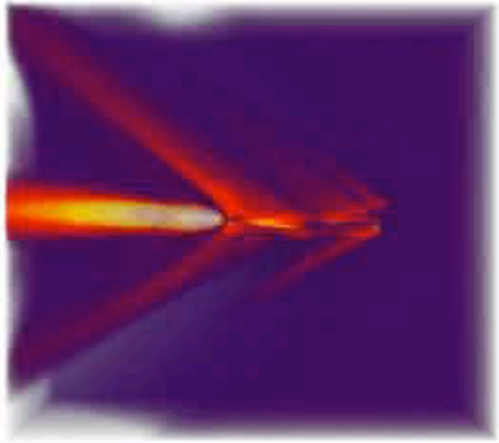
Movie

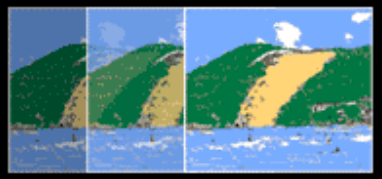




Movie

Fighter
1.40M Tetrahedra

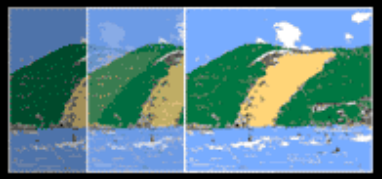




SIBGRAPI 2005

Conclusion

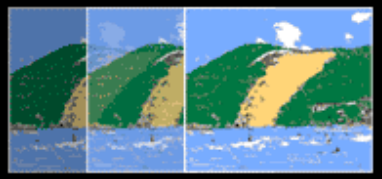
- New sampling approach which simplifies LOD
- Well-suited for a GPU implementation
- Allows dynamic changes to LOD without keeping hierarchical information



SIBGRAPI 2005

Open Research

- Develop techniques to refine datasets to respect a given *k*
- Parallel techniques
- Other *k*-Buffer uses: Isosurfaces, rendering effects, etc.
- Better sampling strategies
- Handle even larger data



SIBGRAPI 2005

Acknowledgements

- Cláudio Silva, João Comba, Milan Ikits, Peter Shirley